GUIDA UFFICIALE ALLA PROGRAMMAZIONE DI

GEOS

Berkeley Softworks

PER COMMODORE 64/64C/128

GUIDA UFFICIALE ALLA PROGRAMMAZIONE DI

GEOS

Berkeley Softworks

Michael Farr

GUIDA UFFICIALE ALLA PROGRAMMAZIONE DI

GEOS

Berkeley Softworks

PER COMMODORE 64/64C/128

SOMMARIO

1 IL SISTEMA OPERATIVO GEOS

Introduzione

Benvenuti alla programmazione in ambiente GEOS. Chi ha già familiarità con l'architettura interna del computer Commodore 64 (C-64), troverà particolarmente semplice iniziare a programmare in ambiente GEOS. Chi invece non ha alcuna esperienza con il C-64, si accorgerà ben presto di compiere rapidi progressi, dal momento che il sistema operativo GEOS (GEOS) è in grado di sollevare il programmatore dai compiti più complicati e gravosi, lasciando che si concentri esclusivamente sul flusso delle operazioni fondamentali che l'applicazione deve svolgere.

Questo volume dà per scontato che il lettore conosca la programmazione in linguaggio macchina, e abbia una buona familiarità con il computer C-64. In ogni caso è consigliabile tenere a portata di mano, per eventuali consultazioni, un manuale sul microprocessore 6510 e una copia della *Commodore 64, Guida di riferimento per il programmatore* (Commodore Italiana).

GEOS è l'acronimo di Graphic Environment Operating System (Sistema Operativo ad Ambiente Grafico). Come suggerisce la sigla, GEOS adotta elementi grafici per semplificare l'interfaccia utente e il sistema operativo. La filosofia di GEOS consiste nel delegare al sistema stesso lo svolgimento della maggior parte delle operazioni più complesse di cui normalmente si fa carico l'applicazione: l'accesso al disco, la manipolazione dei bit dello schermo ad alta risoluzione, l'apertura e la chiusura dei menu, i box di dialogo, la gestione dei dispositivi di Input/Output (I/O) tramite i driver...

I programmatori in grado di sfruttare appieno le opportunità offerte da GEOS, potranno realizzare le loro applicazioni con notevole risparmio di tempo e considerevole aumento di qualità. Alcune sue caratteristiche, come la spaziatura proporzionale

delle fonti carattere o il codice turbo per l'accesso al disco, dovrebbero essere inserite dal programmatore per ogni singola applicazione, con evidente spreco di tempo e di fatica. In ambiente GEOS, invece, queste risorse sono già disponibili, pronte per essere impiegate. Nel tempo necessario per realizzare una piccola routine di gestione dei caratteri in input, il programmatore che utilizza GEOS può dedicarsi alla costruzione di un'interfaccia utente completa e funzionale.

La possibilità d'impiegare routine predefinite per i menu, le finestre o le altre caratteristiche grafiche di GEOS, consente di realizzare applicazioni semplici da usare e con una veste grafica di ottima qualità. GEOS è di notevole aiuto anche per l'utente, perché permette di controllare più o meno nello stesso modo – tramite menu e icone – applicazioni di tipo del tutto diverso.

GEOS rende più vasti gli orizzonti applicativi del C-64. La presenza di un sistema turbo di accesso al disco, ad esempio, permette alle applicazioni di scambiare dati con la memoria di massa molto più rapidamente. Database e altre applicazioni possono manipolare ragguardevoli quantità di informazioni con notevole risparmio di tempo.

Il tempo che occorre per prendere confidenza con un nuovo sistema è un vero e proprio investimento. Ma fin dall'inizio, lo sforzo e le energie necessarie per imparare a orientarsi nell'ambiente di lavoro GEOS avranno come ricompensa una notevole rapidità di realizzazione, anche per procedure molto complesse. Gli obiettivi di GEOS sono semplici: maggiori possibilità per il programmatore, e maggiore semplicità per l'utente. Questo libro fa parte dello sforzo che abbiamo fatto per raggiungere questi obiettivi.

Uniformiamo il linguaggio

Prima di proseguire, è necessario spendere due parole sulle notazioni che saranno utilizzate nel corso del manuale. All'interno di questo libro i riferimenti a costanti, locazioni di memoria, variabili e routine sono ottenuti mediante i loro nomi simbolici. Questa scelta ha lo scopo di facilitare la lettura e aiutare nella memorizzazione, dato che è certamente più facile ricordare parole anziché numeri esadecimali. L'istruzione jsr DoMenu illustra già nel nome una parte del suo significato, e si ricorda meglio di jsr $C151. Gli indirizzi e i valori associati ai nomi simbolici si trovano nelle appendici *Costanti*, *Variabili GEOS globali*, e *Routine*. Si è stabilito convenzionalmente di scrivere i nomi delle costanti tutti in lettere maiuscole (FALSE, TRUE), i nomi delle variabili con l'iniziale minuscola e ogni parola successiva con l'iniziale maiuscola (mouseXPos, mouseData); infine i nomi delle routine con l'iniziale di *ogni* parola maiuscola (DoMenu). Oltre a questi nomi simbolici, abbiamo usato nomi particolari anche per indicare alcune macro istruzioni del compilatore Assembly. Per esempio:

```
LoadB        variabile, valore
```

è una macro istruzione che indica:

```
lda          #valore
sta          variabile
```

In appendice è riportata la lista completa delle macro istruzioni utilizzate in questo testo.

Le caratteristiche fondamentali

Le applicazioni in ambiente GEOS hanno a disposizione le seguenti caratteristiche:

- Menu a scomparsa
- Icone
- Spaziatura proporzionale delle fonti carattere
- Routine di gestione delle stringhe di I/O che utilizzano la spaziatura proporzionale delle fonti carattere
- Box di dialogo
- Libreria completa di routine grafiche
- Libreria completa di routine matematiche
- Gestione multitasking all'interno delle applicazioni
- Elevata velocità d'accesso al disco
- Sistema di suddivisione in pagine dei file
- Set completo di interfacce di comunicazione con diverse stampanti e dispositivi di input

GEOS è un sistema operativo dotato di tutte le qualità che caratterizzano i sistemi operativi grafici per computer di maggiori dimensioni, e la sua parte centrale è il Kernel. Si tratta di un programma residente in memoria e sempre in esecuzione. Contiene tutte le routine di gestione delle finestre, dei menu, delle icone, delle fonti carattere e di quant'altro GEOS è in grado di offrire.

Facciamo un esempio per chiarire la differenza tra sistema operativo e interfaccia utente: deskTop non è parte del Kernel, ma solo una delle possibili applicazioni, come geoPaint o geoWrite. È quindi possibile da parte dei programmatori realizzare applicazioni diverse per la gestione dei file, e lasciare deskTop nel cassetto.

Programmando in ambiente GEOS, la maggior parte del lavoro consiste nella stesura di tavole di dati che definiscono le icone, i menu e le relative routine di servizio. Quando viene attivato un menu o un'icona, il Kernel chiama la routine corrispondente. Benché GEOS sia un sistema operativo sofisticato, il flusso delle operazioni primarie che esegue può essere descritto facilmente.

Qualsiasi input da parte dell'utente che dev'essere gestito dal Kernel di GEOS – come l'apertura di un menu, l'inserimento di un testo o lo spostamento del mouse – viene definito "evento" Il Kernel di GEOS è strutturato in modo da contenere tutte le routine necessarie per analizzare gli eventi. L'applicazione deve definire i menu, le icone e qualsiasi genere di evento tramite tavole di dati, nelle quali a ogni dato è associata una routine di servizio, da eseguire quando l'evento viene attivato dall'utente. Una volta che il Kernel di GEOS entra in possesso delle tavole di dati, è in grado di reagire agli eventi chiamando le routine di servizio associate. Per esempio, supponiamo che il programmatore decida di utilizzare tre icone, e per ognuna di esse crei una routine di servizio che esegue un particolare compito. Perché questa struttura diventi operativa, il Kernel di GEOS deve entrare in possesso di una tavola di dati nella quale siano definite le icone e le routine di servizio associate. Una volta che l'applicazione ha provveduto a questo, il controllo degli input che arrivano dall'utente diventa compito del Kernel che analizza gli eventi basandosi sulla tavola di dati e chiama(†) le corrispondenti routine di servizio. Quindi, a parte il lavoro d'inizializzazione e l'esecuzione delle routine di servizio, il controllo delle operazioni in corso è affidato al Kernel di GEOS. Le stesse routine di servizio fanno uso delle caratteristiche di GEOS per svolgere le loro funzioni.

In realtà l'applicazione potrebbe assumere il pieno controllo dell'hardware del computer, ma nella maggior parte dei casi conviene delegare questo lavoro al Kernel di GEOS.

Ad esempio, invece di segnalare all'applicazione che il pulsante del mouse è stato premuto, il Kernel di GEOS potrebbe concludere, analizzando i diversi movimenti del mouse e la pressione del pulsante, che è stato attivato un evento come l'apertura di un menu. In questo caso alcune routine del Kernel, chiamate "esecutori", analizzano la richiesta dell'utente e chiamano la routine di servizio associata all'evento.

Nel caso dell'evento "menu" appena ipotizzato, il Kernel di GEOS inverte per un attimo la voce selezionata e chiama la routine corrispondente. Questo tipo di interazione è conosciuta come "programmazione a gestione di eventi"

Un evento assume principalmente due forme:

1) l'inizio di un'azione voluta dall'utente
2) l'attivazione di un processo temporizzato definito dall'applicazione.

(†) In questo manuale, a meno che non sia altrimenti specificato, "chiamare una routine" significa eseguire l'istruzione assembly jsr (salta alla subroutine), e "ritornare" significa eseguire l'istruzione assembly rts (ritorna dalla subroutine).

Un processo temporizzato può essere, per esempio, una routine eseguita ogni secondo per aggiornare un orologio. Il programmatore non può limitarsi a realizzare la routine, ma deve anche stabilire con quale intervallo di tempo questa dev'essere eseguita. Il Kernel di GEOS provvederà poi a chiamarla periodicamente.

Quando l'utente non opera nessuna azione in input al sistema, gli unici codici che il processore esegue sono quelli del Kernel di GEOS e i processi temporizzati attivi. Per la maggior parte, le applicazioni sono guidate solo dagli eventi. Il Kernel di GEOS è in grado di gestire il mouse in maniera autonoma, e quindi di capire se il pulsante è stato premuto sopra un'icona, un menu o qualche altra parte dello schermo. Il vettore otherPressVec contiene l'indirizzo della routine da chiamare nel caso che il pulsante venga premuto quando il mouse non si trova né su un menu, né su un'icona. Il vettore keyVector contiene l'indirizzo della routine da chiamare quando viene premuto un tasto della tastiera. In entrambi i casi è il Kernel che, tenendo sotto controllo l'hardware del computer, si accorge della pressione del pulsante in aree dello schermo non convenzionali (aree diverse da icone e menu) o della pressione di un tasto, e chiama le routine previste dal programmatore per gestire eventi non convenzionali. Ad esempio, la routine di gestione della pressione di un tasto della tastiera, potrebbe restituire un buffer contenente i caratteri battuti dall'utente. In un'applicazione come un editor di testi, lo schermo rappresenta parte di una pagina; premere il pulsante del mouse sullo schermo significa quindi selezionare una posizione sulla pagina. Il punto selezionato diventa la nuova posizione sulla pagina dove introdurre un testo o un disegno.

Quando l'utente preme il pulsante del mouse in una parte dello schermo diversa da un'icona o un menu, viene eseguita la routine il cui indirizzo è contenuto in otherPressVec. La routine può accedere alle variabili mouseXPos e mouseYPos per determinare la posizione del mouse. Quando invece si preme un tasto, viene eseguita la routine il cui indirizzo è contenuto in keyVector, e l'applicazione può servirsi della routine di sistema GetNextChar per ricevere i caratteri digitati dall'utente. I vettori otherPressVec e keyVector sono inizializzati a 0 per indicare che ai due eventi non è stata associata nessuna routine. La routine d'inizializzazione delle variabili contenuta in un'applicazione, dovrebbe aggiornare questi due vettori con gli indirizzi delle appropriate routine di servizio, oppure lasciarli a 0 se non ne è stata predisposta nessuna.

La gestione della doppia pressione del pulsante del mouse attraverso otherPressVec

Per doppia pressione del pulsante del mouse si intende la pressione del pulsante due volte in rapida successione. Nell'applicazione deskTop, per mandare in esecuzione un file programma bisogna premere il pulsante del mouse sopra la corrispondente icona

due volte in rapida successione. Descriveremo ora come l'applicazione registra una singola o una doppia pressione sul pulsante del mouse, quando il mouse non si trova su un'area convenzionale. Per quanto riguarda la doppia pressione sulle icone, l'argomento sarà discusso nel capitolo relativo alle icone.

Il Kernel di GEOS ha a disposizione una variabile chiamata dblClickCount. Per gestire la doppia pressione del pulsante del mouse, si adotterà il seguente sistema. La prima volta che il mouse seleziona una parte dello schermo non convenzionale, viene chiamata la routine di servizio il cui indirizzo è contenuto in otherPressVec. Uno dei suoi compiti è quello di controllare il valore contenuto in dblClickCount, e se questo è 0, deve aggiornarlo con il contenuto della costante CLICK_COUNT. Quindi la routine esegue le operazioni previste per una singola pressione sul pulsante del mouse e infine ritorna. A ogni chiamata di interrupt, dblClikCount diminuisce di un'unità, se non è già a 0. Se il pulsante del mouse viene premuto un'altra volta prima che dblClickCount arrivi a 0, la routine di servizio conclude che il pulsante del mouse è stato premuto due volte in rapida successione, e quindi esegue le operazioni associate a tale evento. Il tempo massimo fra le due pressioni, dopo il quale la routine di servizio non riconosce più una doppia pressione del pulsante del mouse, ma due singole pressioni separate, è determinato dalla costante CLICK_COUNT.

Insieme ai vettori otherPressVec e keyVector, le routine di servizio dei menu e delle icone costituiscono gli strumenti primari per creare un'applicazione gestita da eventi. Per aumentare la flessibilità operativa, il Kernel di GEOS prevede anche alcune routine che non operano secondo la struttura a eventi.

Il primo passo

Il primo compito che un'applicazione deve svolgere quando viene mandata in esecuzione da deskTop, è quello di definire i menu, le icone e indicare le routine di servizio per gestire la pressione sul pulsante del mouse in aree non convenzionali e la pressione sui tasti della tastiera da parte dell'utente. Dovrebbe inoltre riconfigurare lo schermo, cancellandolo interamente e disegnando quanto è necessario a caratterizzare l'applicazione dal punto di vista grafico.

Quando l'utente seleziona un file da deskTop per mandarlo in esecuzione, il Kernel di GEOS inizializza il sistema a uno stato di default, carica l'applicazione in memoria e la esegue con un jsr alla routine d'inizializzazione prevista dal programmatore. L'indirizzo è contenuto nel blocco File Header (blocco di testata del file) dell'applicazione, di cui parleremo più avanti. La routine che inizializza l'applicazione contiene le tavole di dati necessarie per definire i menu, le icone e altri eventi. Per trasferire le tavole di dati al controllo del Kernel, la routine deve chiamare apposite routine di sistema che possano leggere e attivare gli eventi. Oltre a questo, deve anche preoccuparsi di disegnare lo schermo secondo le esigenze dell'applicazione. Una volta

che la routine d'inizializzazione ha eseguito queste operazioni, restituisce il controllo al Kernel di GEOS. Il codice principale (main program loop) contenuto nel Kernel riprende il controllo e gestisce, grazie alle tavole di dati, la selezione dei menu, delle icone o di altri eventi da parte dell'utente.

Quando viene selezionato un evento, il Kernel di GEOS chiama la routine associata, definita dall'applicazione. Come accade per la procedura d'inizializzazione, ogni routine di servizio restituisce il controllo al Kernel di GEOS nel momento in cui termina la propria esecuzione.

Riepilogo

Nei paragrafi precedenti sono stati illustrati molti argomenti importanti. Riassumiamoli brevemente: il Kernel di GEOS è un sistema operativo che condivide con l'applicazione la memoria del C-64, e viene eseguito in maniera ciclica. Come tutti i sistemi operativi, controlla la maggior parte delle interazioni del sistema con l'hardware. Quando viene selezionato un evento, come la pressione di un tasto sulla tastiera o l'attivazione di un menu, il Kernel chiama la routine di servizio associata. Questa esegue il processo previsto dall'evento, di solito chiamando le routine di gestione grafica e di gestione testi installate nel Kernel di GEOS, e cedendo poi il controllo allo stesso GEOS. A questo punto il Kernel è pronto per analizzare un nuovo evento da parte dell'utente e chiamare la corrispondente routine di servizio.

Consideriamo per esempio deskTop, una delle possibili applicazioni per la gestione dei file. Quando viene selezionata l'icona corrispondente a un file, deskTop comanda al Kernel di caricare in memoria l'applicazione e cederle il controllo. GEOS carica il file da disco, inizializza il sistema a uno stato di default ed esegue la routine d'inizializzazione della nuova applicazione che ha sostituito in memoria deskTop. Questa routine contiene le tavole di dati che definiscono gli eventi e chiama le routine del Kernel che trasformano gli "eventi definiti" in "eventi operativi". Essa deve anche riconfigurare lo schermo con una nuova veste grafica.

In questo libro tutti gli argomenti citati sono ampiamente analizzati e corredati da sintetici esempi applicativi che utilizzano menu, icone e input di testi. Con le capacità di cui dispone GEOS, un'applicazione può giungere a una prima versione funzionante nel giro di una settimana.

Per dare un'idea intuitiva di come lavora il Kernel di GEOS, nei prossimi paragrafi ne descriveremo l'intera struttura.

La struttura del Kernel di GEOS

Il Kernel di GEOS contiene due livelli di codice in costante esecuzione: MainLoop e InterruptMain.

MainLoop

La routine MainLoop è un lungo codice che viene continuamente eseguito. Questa routine, a ogni ciclo, controlla se è stato attivato un evento dall'utente, ed eventualmente chiama la routine di servizio associata. Cioè controlla se sono arrivati dati in input.dall'utente e ne analizza il significato.

La pressione sul pulsante del mouse può indicare:

- la selezione di un'icona
- l'apertura di un menu
- la selezione della voce di un menu
- l'attivazione di otherPressVec (se il mouse non si trova né su un menu né su un'icona).

La pressione di un tasto della tastiera può indicare:

- un testo in input dall'utente, analizzato dalla routine di servizio associata a keyVector
- un testo per un box di dialogo, analizzato dalle routine del Kernel di GEOS.

L'azzeramento di un lasso di tempo predefinito indica:

- il sistema deve mandare in esecuzione il processo temporizzato associato.

Dopo aver accertato la presenza di un input da parte dell'utente, MainLoop decide cosa fare. Nel caso di un menu le possibilità sono due:

1) dev'essere aperto un sotto-menu. Se per esempio è stato selezionato il menu edit, le diverse voci del sotto-menu devono comparire sullo schermo
2) dev'essere mandata in esecuzione la routine di servizio associata alla voce selezionata. Se per esempio è stata selezionata la voce cut, la routine di servizio deve trasferire una parte di testo.

InterruptMain

La routine InterruptMain contenuta nel Kernel di GEOS serve per gestire il 6510 IRQ interrupt che viene attivato 60 volte al secondo da un segnale di interrupt hardware presente all'interno del C-64. Questo segnale è generato dal clock di sistema. Ogni sessantesimo di secondo il microprocessore (CPU 6510) interrompe l'esecuzione di MainLoop ed esegue la routine InterruptMain, la cui durata è molto inferiore a un sessantesimo di secondo. Il suo compito è solo quello di leggere la situazione hardware del sistema. Così anche nel caso che MainLoop impieghi un tempo molto più lungo del normale per completarsi (ad esempio nel caso che l'applicazione definisca routine di servizio particolarmente elaborate), InterruptMain mantiene l'interazione istantanea con l'hardware: la pressione di un tasto o la richiesta di movimento del mouse da parte dell'utente sono eventi che in questo modo non vengono persi.

InterruptMain salva lo stato della CPU e gli pseudoregistri di cui parleremo fra poco, e interagisce con l'hardware del computer. Aggiorna un buffer interno con i caratteri provenienti dall'input dell'utente, decrementa il timer del sistema (vedere il capitolo dedicato agli interrupt) e i timer associati ai processi temporizzati, aggiorna la posizione del mouse sullo schermo e segnala se il pulsante è stato premuto. Se l'utente preme il pulsante del mouse, InterruptMain imposta un flag il cui stato sarà poi controllato da MainLoop. MainLoop, in base alla posizione del mouse rispetto ai menu e alle icone definite dall'applicazione, decide come operare. Quindi la prima operazione della sequenza che caratterizza un evento inizia sempre in InterruptMain. Il mouse e la tastiera sono costantemente gestiti da InterruptMain, e quando avviene qualche cambiamento, è questa routine a impostare gli opportuni flag per segnalare e descrivere l'inizio dell'evento. Questi flag sono controllati da MainLoop almeno una volta per ogni ciclo. InterruptMain ripristina lo stato della CPU e degli pseudoregistri precedentemente salvati, e restituisce il controllo a MainLoop. MainLoop prende visione dei cambiamenti segnalati da InterruptMain e chiama le appropriate routine di servizio. Mentre InterruptMain una volta iniziata non può essere interrotta, e quindi viene eseguita sempre per intero, MainLoop, come ogni altra routine definita dall'applicazione, si interrompe ogni sessantesimo di secondo perché InterruptMain possa sempre tenere aggiornato lo stato dell'hardware.

La maggior parte dei programmatori del C-64 sono abituati a creare versioni personali delle routine MainLoop e InterruptMain per le loro applicazioni, ma è improbabile che tali routine siano così flessibili da poter essere utilizzate più di una volta. Utilizzando GEOS, invece, questo lavoro impegnativo e ripetitivo non dev'essere svolto. Il Kernel di GEOS è simile a un blocco di creta che attende di essere modellato dal programmatore. Le applicazioni GEOS compatibili sono costituite da una collezione di tavole per definire i dati necessari alla gestione degli eventi. Il flusso dei controlli sugli eventi è strutturato dal Kernel di GEOS.

Ogni volta che una routine di servizio effettua un ritorno (rts), restituisce il controllo a MainLoop. Una routine di servizio può ridisegnare lo schermo, inizializzare nuovamente gli eventi, le icone, i menu e così via, e alla fine restituire tranquillamente il controllo a MainLoop che riprende il suo ciclo esattamente dal punto in cui era stato interrotto dalla chiamata della routine. MainLoop continua a girare come se niente fosse successo, ma l'ambiente dove si trova a operare può essere stato completamente riconfigurato. Per esempio, alla voce di un menu può essere associata una routine di servizio che cambia completamente l'ambiente con il quale l'utente interagiva, creandone uno del tutto nuovo. Di norma, una delle prime operazioni eseguite da MainLoop consiste nel controllare se è stata selezionata un'icona. Ma se c'è un menu aperto questo controllo non viene effettuato, in quanto un menu e un'icona non possono essere selezionati contemporaneamente. Nel successivo ciclo di MainLoop, i nuovi menu, le nuove icone o altri eventi saranno a loro volta analizzati.

Affidare a GEOS la maggior parte del lavoro più complicato e ripetitivo consente al programmatore di non reinventare per ogni applicazione un nuovo castello di routine.

Il Kernel di GEOS contiene le più svariate routine, e permette di creare applicazioni anche molto complesse, come geoWrite, geoPaint e deskTop, che utilizzano appieno le risorse messe a disposizione dal sistema. Il prossimo paragrafo spiega come chiamare le routine del Kernel.

L'esecuzione delle routine del Kernel

Questo paragrafo descrive la procedura che il programmatore deve seguire per chiamare le routine del Kernel di GEOS. La prima convenzione adottata nella creazione del Kernel di GEOS è stata quella di allocare alcune variabili in pagina zero. In questo modo, infatti, le istruzioni del 6510 che indirizzano la pagina zero sono più rapide. Inoltre, per un'altra convenzione di base, le routine del Kernel di GEOS utilizzano queste variabili per accettare parametri, effettuare calcoli interni e restituire valori. Configurare le routine con particolari variabili di input e di output dei parametri permette di controllare con estrema semplicità quali cambiamenti vengono introdotti in memoria da ogni routine. Inoltre queste convenzioni permettono ai programmatori, e non solo alla Berkeley Softworks, di utilizzare le routine del Kernel con relativa facilità.

A questo proposito, 32 byte in pagina zero, allocati a partire dalla locazione $02, sono riservati per essere impiegati come pseudoregistri. Queste locazioni di memoria, divise in 16 word, sono numerate da r0 a r15. Il byte basso di ogni pseudoregistro può essere indicato sia come rN sia come rNL, dove N è il numero del registro. Il byte alto dev'essere indicato come rNH.

Normalmente i parametri per le routine vengono passati attraverso gli pseudoregistri. Le routine utilizzano gli pseudoregistri per effettuare i loro calcoli interni. Anziché tentare subito di gestire centinaia di locazioni del Kernel di GEOS, il programmatore inizia manipolandone solo 16. Dal momento che InterruptMain, prima di procedere alla propria esecuzione, salva lo stato di tutti gli pseudoregistri (da r0 a r15), può tranquillamente chiamare le routine del Kernel senza preoccuparsi che le stesse routine siano utilizzate in quel momento anche da MainLoop o dall'applicazione. In questo modo InterruptMain, che sfrutta le potenzialità offerte dalle routine di sistema senza intaccare lo svolgimento di MainLoop e delle applicazioni, può diventare anche molto sofisticata.

Gli pseudoregistri non sono l'unico modo per trasferire parametri alle routine del Kernel. Qualche volta, per migliorare la velocità, possono essere usati i registri a, x, y e anche il flag carry. Ma c'è ancora un altro modo per comunicare dati alle routine: la chiamata *inline*. Una chiamata inline risolve il problema delle routine che devono essere eseguite di frequente e che richiedono il trasferimento di numerosi parametri. Normalmente si dovrebbero utilizzare molte istruzioni del 6510 in linguaggio macchina per inizializzare gli pseudoregistri con i dati da trasferire, ma questo procedimento, oltre che rallentare l'esecuzione della chiamata, spreca spazio di

memoria. Per ovviare all'inconveniente, alcune routine che richiedono numerosi parametri possono essere chiamate anche utilizzando la chiamata inline. Mentre la chiamata normale di una routine prevede che i parametri siano trasferiti attraverso gli pseudoregistri, la versione inline fa in modo che la routine accetti i parametri prelevandoli dai byte che seguono immediatamente l'istruzione jsr di chiamata. Per esempio, la chiamata inline per disegnare un rettangolo è la seguente:

```
jsr     i_Rectangle  ;disegna un rettangolo con la matrice grafica corrente (la matrice
                     ;grafica corrente puo' essere cambiata con la routine SetPattern)
.byte   0            ;lato superiore del rettangolo. Range possibile: 0-199
.byte   199          ;lato inferiore del rettangolo. Range possibile: 0-199
.word   0            ;lato sinistro del rettangolo. Range possibile: 0-319
.word   319          ;lato destro del rettangolo. Range possibile: 0-319
```

Mentre la chiamata normale dovrebbe essere:

```
LoadB   r2L,0        ;lato superiore del rettangolo. Range possibile: 0-199
LoadB   r2H,199      ;lato inferiore del rettangolo. Range possibile: 0-199
LoadW   r3,0         ;lato sinistro. Range possibile: 0-319
LoadW   r4,319       ;lato destro. Range possibile: 0-319
jsr     Rectangle    ;disegna il rettangolo
```

Quando viene chiamata una routine inline, questa preleva subito una word dallo stack. Di solito, in condizioni non inline, questa word individua l'indirizzo di ritorno che il processore memorizza nel registro program counter (PC) quando incontra l'istruzione rts al termine della routine. Invece, in condizioni inline, la word individua l'inizio del gruppo di parametri che seguono immediatamente la chiamata. La routine inline preleva il numero di byte corrispondente al numero di parametri di cui ha bisogno e aggiorna una word con l'indirizzo corrispondente alla prima locazione di memoria dopo il gruppo di dati. La word viene reinserita nello stack per aggiornare l'indirizzo di ritorno della routine, e solo dopo queste operazioni la routine inizia a svolgere i compiti per cui è stata programmata.

L'impiego della chiamata inline è consigliato nel caso di routine che devono essere chiamate di frequente e che necessitano di numerosi parametri di valore fisso. Per esempio, per cancellare una parte dello schermo, conviene effettuare una chiamata inline a i_Rectangle anziché una chiamata normale a Rectangle qualora i parametri di input non debbano variare. In questo modo si utilizzano meno byte che per una chiamata normale, perché quest'ultima dovrebbe essere preceduta dall'aggiornamento degli pseudoregistri tramite le istruzioni in linguaggio macchina del 6510. Per essere precisi, la macro istruzione LoadW r3,0 occupa otto byte di memoria mentre la direttiva .word 0 ne occupa solo due. I nomi delle routine inline iniziano sempre con i_, come i_Rectangle.

Sono le routine del Kernel di GEOS che utilizzano numerosi parametri di input hanno a disposizione anche la chiamata inline. Le routine utilizzate meno frequentemente o quelle che utilizzano uno o due parametri, prevedono solo la chiamata normale. Anche il programmatore può facilmente installare, all'interno delle sue applicazioni, alcune routine inline.

In questo paragrafo abbiamo descritto in che modo le applicazioni possono effettuare una chiamata alle routine del Kernel, e come quest'ultimo utilizza le routine di servizio delle applicazioni. Abbiamo descritto quali sono i compiti di MainLoop e Interrupt-Main nel Kernel di GEOS. Nei prossimi paragrafi spiegheremo in che modo un'applicazione può inserire i propri codici all'interno di InterruptMain e MainLoop. Di solito questa procedura è da sconsigliare, ma in alcune circostanze diventa necessaria per gestire particolari dispositivi hardware esterni. Se si dovesse incontrare questa necessità, l'applicazione può aggiornare alcuni vettori previsti nella RAM di sistema, che permettono di aggiungere codici particolari a InterruptMain o a MainLoop. Una buona regola è evitare di alterare i codici di MainLoop o InterruptMain. Infatti una routine di interrupt in un'applicazione può incontrare qualche difficoltà nel mantenere il sincronismo fra MainLoop e InterruptMain.

Codice non strutturato in eventi

La maggior parte delle applicazioni non ha bisogno di ricorrere a codici non gestiti da eventi. Questi codici devono essere eseguiti a ogni chiamata di interrupt o a ogni ciclo di MainLoop, indipendentemente dalle azioni dell'utente, e non possono essere predisposti come processi temporizzati. L'unico motivo per cui il programmatore può avere la necessità di crearli, è la gestione di speciali dispositivi hardware. Alterando particolari vettori di sistema opportunamente predisposti, il programmatore può richiedere al Kernel di GEOS di controllare alcune routine non gestite da eventi. I vettori per aggiungere codici a InterruptMain o a MainLoop sono intTopVector, intBotVector, e appMain. Se un'applicazione esige che la sua routine di interrupt sia eseguita prima dei codici di InterruptMain, può alterare l'indirizzo contenuto in intTopVector(†). In questo caso viene eseguito un salto indiretto attraverso intTopVector, che normalmente contiene l'indirizzo di InterruptMain.

Aggiornando questo vettore con l'indirizzo di una routine personalizzata, questa viene eseguita prima di ogni altra cosa a ogni interrupt. La fine della routine di interrupt dell'applicazione dev'essere costituita dall'istruzione "jmp InterruptMain" Allo stesso modo, per mandare in esecuzione un codice addizionale alla fine della routine InterruptMain, si deve alterare intBotVector. Alla fine dei codici di

(†) Gli indirizzi reali di memoria delle variabili e delle routine, e i valori delle costanti citate nel testo, sono riportati in appendice.

InterruptMain, il Kernel di GEOS effettua una chiamata alla subroutine indirizzata dal vettore intBotVector, a meno che questo non sia zero (che è il suo valore di default). Qualsiasi routine eseguita attraverso intBotVector deve terminare con l'istruzione rts, e non rti.

La maggior parte della programmazione può essere compiuta utilizzando la gestione a eventi. È comunque possibile aggiungere routine addizionali a MainLoop, aggiornando il vettore appMain con l'indirizzo della routine da chiamare. Durante ogni ciclo di Mainloop viene eseguito un salto indiretto attraverso appMain, a meno che questo non sia zero (che è il suo valore di default). Un rts alla fine della routine chiamata attraverso appMain, restituisce il controllo alla routine MainLoop del Kernel di GEOS.

Procedura consigliata per la stesura di un'applicazione in ambiente GEOS

Siamo ora in grado di analizzare i passi necessari per programmare in ambiente GEOS.

1) *Scelta degli eventi*: i menu, le icone o gli altri eventi che l'applicazione deve definire. Un tipo speciale di eventi è costituito dai processi temporizzati che saranno descritti nei prossimi capitoli.

2) *Definizione degli eventi*: aggiornamento dei vettori e preparazione delle tavole di dati che definiscono gli eventi (per esempio la struttura di un menu si realizza definendo una semplice tavola di dati associata).

3) *Stesura delle routine*: creazione delle routine di servizio eseguite da MainLoop per reagire agli eventi previsti dal programmatore.

Per meglio comprendere l'importanza dei passi che abbiamo qui elencato, vediamo cosa significa programmare in ambiente GEOS. GEOS permette di trasformare un'idea in un'applicazione funzionante in brevissimo tempo, in quanto aiuta il programmatore a suddividere il lavoro in tanti piccoli moduli contenenti tavole di dati e le relative routine di servizio associate agli eventi. Illustriamo ora brevemente la configurazione hardware creata da GEOS: il modo grafico utilizzato, la sua allocazione in memoria e come sono impostati i registri di selezione dei banchi di memoria.

È possibile programmare in ambiente GEOS senza conoscere niente dei modi grafici e della selezione dei banchi di memoria. Chi è nuovo alla struttura hardware del C-64 non deve preoccuparsi se i prossimi paragrafi gli sembreranno di difficile comprensione. Si dà comunque per scontato che il lettore abbia consultato il volume *Commodore 64, Guida di riferimento per il programmatore*. È improbabile che il programmatore

abbia necessità di cambiare la mappa di memoria (Memory Map) standard di GEOS. Può comunque accadere che l'applicazione debba accedere alla ROM del Kernel di cui la Commodore ha dotato il C-64, oppure a una routine di calcolo in virgola mobile nella ROM del Basic, e successivamente ripristinare la configurazione standard per operare normalmente. Il resto di questo capitolo è dedicato all'analisi della struttura hardware configurata da GEOS. Questo significa illustrare il modo grafico prescelto, le aree di memoria occupate dal Kernel, le differenze tra le varie versioni del Kernel residenti in memoria, l'impostazione dei registri hardware del sistema e infine il metodo per alterare la configurazione della memoria e accedere alle routine contenute nella ROM del Kernel, o del Basic, del C-64.

Modo bit-map in alta risoluzione

GEOS utilizza il modo grafico bit-map del C-64 con una risoluzione di 320 x 200 pixel. In questo modo sono occupati 8000 byte (200 linee di scansione da 40 byte per linea) per gestire lo schermo in alta risoluzione.

Per facilitare la gestione dello schermo grafico in ambiente GEOS, viene allocato in memoria un buffer di altri 8000 byte, il *buffer di schermo*, la cui funzione è quella di conservare una copia dello schermo grafico corrente. Tramite opportune routine, le immagini (parti di schermo) memorizzate nel buffer vengono copiate sullo schermo, e viceversa. Avere una copia dello schermo grafico è uno stratagemma molto utile per gestire menu o box di dialogo sullo schermo corrente. In pratica le parti di schermo che vengono cancellate dall'apertura di un menu o dall'apparizione di un box di dialogo sono preventivamente salvate nel buffer. Per ripristinare sullo schermo l'immagine sottostante, per esempio alla chiusura di un menu, le routine di gestione dei menu e dei box di dialogo, indipendentemente dall'applicazione, copiano sullo schermo i dati salvati nel buffer. In questo modo l'applicazione non deve preoccuparsi di ripristinare l'immagine coperta da un menu, operazione che in certi casi potrebbe risultare impossibile.

Le routine di recupero delle immagini nascoste dai menu e dai box di dialogo, possono essere utilizzate anche dalle routine di servizio delle applicazioni. L'applicazione geoPaint utilizza queste routine per l'opzione undo, la quale ripristina la pagina grafica eliminando gli effetti dell'ultima operazione. Le routine del Kernel di GEOS realizzate per gestire le copie conservate nel buffer di schermo, includono RecoverAllMenus, RecoverLine, RecoverMenu e RecoverRectangle. Queste routine sono illustrate nei capitoli dedicati alla grafica e ai menu. L'impiego del buffer di schermo, da parte del Kernel di GEOS, può essere disabilitato dall'applicazione, se quest'ultima desidera utilizzare l'area di memoria da esso occupata per usi diversi. Questo argomento è analizzato nel capitolo "GEOS e la grafica".

La mappa di memoria

La mappa di memoria del Kernel di GEOS indica com'è organizzata la memoria del C-64 e quali sono gli spazi liberi per le applicazioni. Le applicazioni hanno a disposizione circa 22K di memoria, dall'indirizzo $0400 all'indirizzo $5FFF. Con speciali accorgimenti, le applicazioni possono estendere tale spazio utilizzando anche il buffer di schermo. Questa operazione libera altri 8K, aumentando lo spazio disponibile a 30K. È una quantità di memoria che può apparire limitata per la stesura di applicazioni complesse, ma è importante non dimenticare che i codici di gestione dei menu, delle icone, dei box di dialogo, dei vari buffer e del disco sono già presenti nel Kernel di GEOS. Questo significa diminuire considerevolmente il tempo e gli sforzi necessari ai programmatori per realizzare le applicazioni, e che 22K o 30K di memoria sono più che sufficienti. Inoltre l'elevata velocità operativa delle routine di accesso al disco contenute nel Kernel di GEOS permette di interscambiare con grande rapidità moduli di programma memorizzati sul disco. In questo modo l'applicazione, a seconda del tipo di evento, può velocemente caricare da disco il modulo di programma corrispondente, ed eseguirlo. Grazie a questa gestione in overlay, le applicazioni in ambiente GEOS possono anche occupare uno spazio maggiore della memoria a loro disponibile, e quindi diventare molto complesse. Per facilitare la gestione dei moduli di programma da parte del modulo principale dell'applicazione, è stata ideata e realizzata la struttura dei file VLIR. A questo argomento, per l'importanza che riveste, abbiamo dedicato un intero capitolo.

Per un programmatore, generalmente, le uniche informazioni da conoscere sulla mappa di memoria del Kernel di GEOS sono la memoria disponibile per le applicazioni e le locazioni disponibili per memorizzare i dati.

La RAM disponibile per i dati (oltre allo spazio riservato all'applicazione, che il programmatore potrebbe sfruttare anche per i dati), è suddivisa in tre aree separate. La prima consiste negli pseudoregistri r0 - r15. Per utilizzarli, bisogna ricordare che anche le routine del Kernel ne fanno uso per i loro calcoli interni. Saranno date ampie informazioni sugli pseudoregistri utilizzati da ogni routine del Kernel di GEOS. La seconda area è costituita da 4 byte dalla locazione $00FB alla $00FE in pagina zero, che non sono utilizzati né dal Basic, né dal Kernel del C-64. In ambiente GEOS sono chiamati pseudoregistri a0 e a1. Impiegando gli pseudoregistri, che sono tutti allocati in pagina zero, il programmatore e le routine del Kernel di GEOS risparmiano byte di memoria. Infatti, l'indirizzamento in pagina zero richiede un solo byte, oltre al codice dell'istruzione. Tale risparmio di memoria diventa importante, per esempio, quando si devono aggiornare numerosi pseudoregistri con i parametri da passare a una routine.

C'è un'altra regione di memoria utilizzabile dall'applicazione per i suoi dati, che si estende dalla locazione $70 alla $7F. Queste locazioni di memoria sono gli pseudoregistri a2 - a9. Infine è disponibile all'applicazione lo spazio di memoria compreso fra l'indirizzo $7F40 e l'indirizzo $7FFF, anche se si tratta di aree di memoria

non contenute in pagina zero. Per esempio, l'applicazione deskTop utilizza il vettore di byte da $7F40 a $7FFF per memorizzare sequenzialmente i gruppi di dati che caratterizzano ogni icona. Per un panorama completo dello spazio di memoria disponibile all'applicazione, consultare la Memory Map in appendice, o la seguente tavola riassuntiva.

La mappa di memoria di GEOS

Numero byte	Range di indirizzo	Descrizione
1	$0000	Registro direzione dati del 6510
1	$0001	Registro di I/O del 6510
110	$0002-$006F	Pagina zero per GEOS e per le applicazioni
16	$0070-$007F	Pagina zero solo per le applicazioni
123	$0080-$00FA	Pagina zero per le routine del Kernel e del Basic del C-64
4	$00FB-$00FE	Pagina zero solo per le applicazioni
1	$00FF	Usata dal Kernel e dal Basic del C-64
256	$0100-$01FF	Stack del 6510
512	$0200-$03FF	RAM usata dal Kernel del C-64
23552	$0400-$5FFF	RAM per le applicazioni e i dati
8000	$6000-$7F3F	RAM per il buffer di schermo
192	$7F40-$7FFF	RAM per i dati delle applicazioni
2560	$8000-$89FF	Buffer di gestione del disco e variabili di GEOS
512	$8A00-$8BFF	Dati degli sprite
1000	$8C00-$8FE7	Matrice dei colori dello schermo
16	$8FE8-$8FF7	RAM per GEOS
8	$8FF8-$8FFF	Puntatori agli sprite
4096	$9000-$9FFF	Codici del Kernel di GEOS
8000	$A000-$BF3F	Schermo in Hi-Res di GEOS o ROM del Basic
192	$BF40-$BFFF	Tavole di dati di GEOS
4096	$C000-$CFFF	4K di codice del Kernel di GEOS sempre residente
4096	$D000-$DFFF	4K di codice del Kernel di GEOS o 4K di spazio per I/O del C-64
7808	$E000-$FE7F	8K di codice del Kernel di GEOS o 8K di ROM del Kernel del C-64
378	$FE80-$FFF9	Driver di input
6	$FFFA-$FFFF	6510 NMI, IRQ, e vettori di reset

Il controllo degli I/O, dei disegni sullo schermo e degli interrupt, può e dev'essere effettuato dal Kernel di GEOS. Le routine del Kernel sono semplici da usare e, che l'applicazione le impieghi o no, occupano spazio di memoria. I prossimi due paragrafi illustrano nei dettagli la configurazione hardware utilizzata da GEOS e possono essere tralasciati dalla maggior parte dei lettori. Sono stati inclusi nel libro per offrire ai programmatori la possibilità di gestire dispositivi di I/O che il Kernel di GEOS non supporta (per ora), e la possibilità di usare il Basic al posto del linguaggio macchina.

La versione del Kernel

All'interno del Kernel di GEOS ci sono diversi byte utilizzati per identificare la versione di GEOS correntemente in memoria. Alla locazione $C006 si trova la stringa "GEOS BOOT". Questa stringa può essere utilizzata per determinare se l'applicazione è stata caricata da GEOS. I programmatori che non utilizzano le routine del Kernel di GEOS nelle loro applicazioni possono scrivere in tutta la RAM, tranne che nel vettore di dati che va da $C000 a $C07F. Questo vettore contiene i codici che le applicazioni devono usare per riattivare GEOS; può essere spostato in qualunque zona, ma per mandarlo in esecuzione si deve riportarlo a $C000.

Immediatamente dopo la stringa "GEOS BOOT" ci sono due nibble (che insieme formano un byte) contenenti il numero della versione. Normalmente questo byte contiene $12 o $13 per indicare rispettivamente la versione 1.2 o la 1.3. Nel Kernel di GEOS versione 1.3 sono presenti informazioni aggiuntive subito dopo il byte della versione corrente. Il primo byte indica la lingua della versione. Il secondo byte non viene impiegato, mentre il terzo byte, sysFlgCopy, viene impiegato da GEOS V1.3 per memorizzare lo stato del sistema quando viene riceduto il controllo al Basic. I cinque byte seguenti sono riservati per future espansioni e sono attualmente a $00. Al momento in cui scriviamo, sono disponibili le versioni in inglese, tedesco, francese, olandese e italiano, mentre in futuro sono previste anche versioni in lingua svedese, spagnola e portoghese.

Questo vettore viene illustrato nella tavola della pagina successiva.

Byte d'informazione del Kernel di GEOS

```
.psect  $C000          ;Il codice del Kernel inizia a $C000

BootGEOS:
  jmp   o_BootGEOS     ;Salta al vettore di rientro in GEOS. Se la routine o_BootGEOS
                       ;in future versioni si trovera' a un diverso indirizzo, bastera'
                       ;operare un salto a BootGEOS ($C000) per risolvere il problema.
                       ;Infatti nel passaggio da una versione a un'altra lo spazio da
                       ;$C000 a $C07F viene conservato, e saltando a $C000 si ottiene
                       ;il caricamento di GEOS
ResetHandle:
  jmp   o_ResetHandle  ;Questo e' un vettore di salto utilizzato da GEOS durante la
                       ;procedura di Cold Start. Le applicazioni non possono
                       ;chiamare questa routine
BootFileName:
  .byte "GEOS BOOT"    ;L'indirizzo e' $C006. Questa stringa puo' essere
                       ;utilizzata per determinare se un'applicazione e' stata
                       ;caricata da GEOS. Si puo' verificarne la presenza anche con
                       ;un'applicazione in Basic: in caso positivo si puo' mandare in
                       ;esecuzione l'istruzione sys (49152) per ricaricare GEOS
Version:
  .byte $13            ;Questo byte contiene il numero della versione.
                       ;Le versioni correnti sono la 1.2 e la 1.3

                       ;Quanto segue e' presente solo nella versione 1.3

Nationality:
  .byte x              ;Byte che indica la lingua usata dal sistema:
                       ;INGLESE    =   0
                       ;TEDESCO    =   1
                       ;FRANCESE   =   2
                       ;OLANDESE   =   3
                       ;ITALIANO   =   4
                       ;SVEDESE    =   5      (non installata)
                       ;SPAGNOLO   =   6      (non installata)
                       ;PORTOGHESE =   7      (non installata)
futureUse1:
  .byte 0              ;$C011 Riservato per impieghi futuri
```

SEGUE

SEGUE

```
sysFlgCopy:
  .byte 0            ;$C012 Questo flag viene impiegato solo da GEOS V1.3 quando
                     ;e' presente un'espansione RAM
                     ;Bit 7: se e' impostato a 1, MoveData in GEOS impiega l'espan-
                     ;sione RAM per accelerare gli spostamenti dei blocchi di dati
                     ;Bit 6: se e' impostato a 1, $08300-$0B8FF mantiene i driver
                     ;per i drive dall'A al C
                     ;Bit 5: se e' impostato a 1, saltando a $C000 si produce
                     ;la riattivazione del sistema dall'espansione RAM (il disco GEOS
                     ;di caricamento non dev'essere necessariamente nel drive)

  .byte 0            ;Riservato per usi futuri
  .byte 0            ;Riservato per usi futuri
  .byte 0            ;Riservato per usi futuri
  .byte 0            ;Riservato per usi futuri
  .byte 0            ;Riservato per usi futuri
```

Configurazioni dei banchi di memoria e del sistema

La maggior parte del Kernel di GEOS si trova in memoria da $BF40 in poi. Occupa uno spazio nella RAM normalmente utilizzato per altri scopi. Lo spazio da $D000 a $DFFF è impiegato per gli I/O, ma il C-64 offre la possibilità di sovrapporgli della RAM utilizzabile per altri scopi. Nello stesso modo le ROM del Basic e del Kernel del C-64 possono essere sovrapposte da aree di 8K di RAM libera. Durante operazioni normali, tutti i banchi di memoria in cui sono contenuti i codici del Kernel di GEOS sono sovrapposti alla ROM del Basic, alla ROM del Kernel e allo spazio di I/O del C-64. Tutti gli I/O sono gestiti dal Kernel di GEOS che si preoccupa di selezionare i banchi di memoria durante l'esecuzione delle routine di interrupt o di accesso al disco.

La selezione dei banchi viene effettuata tramite il registro allocato a $0001 e da due linee provenienti dalla porta predisposta per l'inserimento di una cartuccia d'espansione. Dal momento che il Kernel di GEOS non utilizza alcuna cartuccia, le resistenze di pull up delle due linee interne al C-64 impostano lo stato delle linee ad "alto". Il registro che controlla la memoria di schermo e la memoria carattere è $D018.

Se l'applicazione si trova nella necessità di accedere allo spazio di I/O senza poter utilizzare le apposite routine del Kernel di GEOS, o vuole accedere alla ROM del Kernel o del Basic del C-64, deve eseguire le due routine del Kernel di GEOS InitForIO e DoneWithIO. Queste routine, rispettivamente, salvano e ripristinano la configurazione della mappa di memoria utilizzata da GEOS, e controllano gli interrupt e gli sprite a seconda delle necessità.

Segue una tavola che illustra come normalmente GEOS aggiorna i registri che configurano il sistema.

Impostazione dei registri di controllo effettuata da GEOS

Funzione di controllo	Indirizzo	Valore	Descrizione
Selezione banchi	$0001	xxxx000x	Seleziona quale banco di ROM deve apparire nello spazio indirizzabile. GEOS disabilita il Kernel del C-64, lo spazio di I/O e la ROM del Basic
Selezione del banco visto dal VIC	$DD00	xxxxxx01	Seleziona quale area da 16K e' vista dal VIC. GEOS seleziona il banco 2 da $8000 a $BFFF
Memoria di schermo	$D018	0011xxxx	Determina la locazione della memoria di schermo. GEOS seleziona l'area di memoria che va da $8C00 a $8FE7
Memoria carattere	$D018	xxxx010x	Seleziona dove appare la memoria carattere vista dal VIC nel banco selezionato. GEOS la alloca da $A000 a $BF3F

Costanti per selezionare i banchi

```
IO_IN            = $35      ;60K RAM, 4K di spazio I/O attivati
RAM_64K          = $30      ;64K RAM di sistema
KRNL_BAS_IO_IN   = $37      ;ROM del Kernel e del Basic attivate
KRNL_IO_IN       = $36      ;ROM del Kernel e spazio I/O attivati
```

Direttive del compilatore Assembly

L'ambiente di sviluppo che abbiamo utilizzato alla Berkeley Softworks può essere diverso dal vostro. Il compilatore che adoperiamo è di nostra crezione. Nelle semplici applicazioni che presentiamo in questo volume riportiamo i listati generati dal nostro compilatore, all'interno dei quali compaiono le nostre direttive e macro istruzioni. Abbiamo cercato di utilizzare le macro istruzioni il meno possibile, per non ostacolare la comprensione dei listati. In appendice ne è riportata l'intera documentazione. Segue una tavola che illustra le direttive del nostro compilatore.

Direttive del compilatore utilizzate negli esempi

Tipo	Pseudocodice direttiva	Argomento	Descrizione
Imposta l'indirizzo:			
	.psect	[VALORE]	VALORE viene utilizzato come indirizzo. Il codice che segue e' compilato a partire dall'indirizzo VALORE. Se VALORE non e' specificato, la compilazione genera un codice che il linker puo' rilocare
	.ramsect	[VALORE]	Stesso significato di .psect, ma per le variabili
Label:			
	NOME:		Assegna l'indirizzo corrente a NOME. I due punti sono opzionali se la label non viene spostata dal margine
Costanti:			
	NOME	=[carattere]VALORE	Eguaglia NOME a VALORE, dove VALORE e' un numero decimale se non e' preceduto da un carattere; un numero esadecimale e' sempre preceduto dal simbolo $, mentre un numero binario e' preceduto da %
Dati:			
	.byte	val1, val2...	Memorizza val1, val2... in byte sequenziali

SEGUE

SEGUE

```
        .word           val1, val2...       Memorizza val1, val2... in word
                                            sequenziali da 16 bit
RAM:
        .block          VALORE              Alloca il VALORE di byte nella memoria
                                            Normalmente segue una label
Compilazione condizionale:
        .if             espressione         Se "espressione" e' vera, compila
                                            il codice incluso
        .elif                               Termina una parte di codice iniziata con
                                            if e ne inizia un'altra
        .else                               Termina una parte di codice iniziata con
                                            if e ne inizia un'altra alternativa alla
                                            precedente
        .endif                              Termina qualunque struttura aperta
```

La prima realizzazione

Nel prossimo capitolo si inizia a descrivere passo per passo la programmazione in ambiente GEOS. Il primo esempio è una piccola applicazione che contiene solo una routine d'inizializzazione per pulire lo schermo e definire alcune icone. Ogni icona, quando è selezionata, chiama una routine di servizio che cancella lo schermo e definisce una struttura di menu. Cercheremo così di descrivere come devono essere definite le icone e i menu in ambiente GEOS.

Dopo questo esempio applicativo, spiegheremo come realizzare l'applicazione, memorizzarla su disco ed eseguirla con GEOS.

Nei capitoli successivi parleremo della grafica, dei testi, dei file, dei box di dialogo, degli interrupt, della stampa e degli sprite. Ogni capitolo contiene una spiegazione generale, seguita da un esempio, e quindi una completa descrizione delle routine del Kernel inerenti all'argomento illustrato.

Più avanti spiegheremo come realizzare un driver di input e uno di stampa, utilizzando la libreria di routine installata nel Kernel di GEOS. Finché non si raggiunge una completa familiarità con l'ambiente di GEOS, è spesso utile servirsi di routine di servizio fittizie che eseguono solo un rts. In questo modo la struttura dei menu e delle icone può essere verificata prima che vengano aggiunte le routine di servizio associate. Dopo che gli eventi sono stati definiti, i menu si aprono e le icone si invertono momentaneamente, anche se le routine di servizio non eseguono assolutamente

niente. In questo modo il programmatore può collaudare la struttura generale dell'applicazione, prima di addentrarsi nella stesura dei codici di servizio degli eventi.

La compatibilità delle applicazioni con GEOS 128

In linea di massima, le applicazioni create per GEOS 64 che sfruttano la jump table a $C100 e soprattutto delegano al sistema operativo lo svolgimento di tutte le funzioni di basso livello, non dovrebbero incontrare problemi di compatibilità se vengono mandate in esecuzione con GEOS 128. La compatibilità è possibile dal momento che GEOS 128 è un ampliamento di GEOS 64, e come tale ne conserva interamente le caratteristiche. Le variabili globali di sistema possedute sia da GEOS 128 sia da GEOS 64 non presentano alcuna differenza; tutte le routine del Kernel svolgono gli stessi compiti in entrambi i sistemi, con l'unica differenza che GEOS 128 ne aggiunge diverse altre. In particolare, GEOS 128 si avvicina molto alla struttura di GEOS V1.3, dal momento che è in grado di gestire le espansioni RAM nello stesso modo. Vedremo comunque quali potrebbero essere le ragioni di eventuali incompatibilità e come porvi rimedio. Per adesso è sufficiente sottolineare che in linea di massima tutte le applicazioni realizzate seguendo le direttive di questo manuale, dovrebbero funzionare correttamente anche con GEOS 128 nel modo a 40 colonne.

Nel momento in cui scriviamo, non sono state ancora realizzate applicazioni per GEOS 64 che siano in grado di sfruttare il modo a 80 colonne offerto da GEOS 128 e la frequenza di clock di 2 MHz. Questo non significa però che ciò non sarà possibile in seguito. Qualsiasi applicazione realizzata per GEOS 64, come vedremo, può installare nel menu GEOS la voce switch 40/80, mettendo così a disposizione dell'utente uno schermo a risoluzione orizzontale duplicata (640 x 200).

Per non creare inutili confusioni, nel corso del manuale faremo riferimento esclusivamente a GEOS 64, e dedicheremo il capitolo 21 alla trattazione di tutti gli argomenti utili al programmatore che voglia creare applicazioni compatibili con GEOS 128. In quel capitolo si darà per scontato che il lettore abbia già letto l'intero manuale e sia quindi a conoscenza delle caratteristiche fondamentali del sistema operativo. Le appendici A e B, oltre a descrivere tutte le costanti e le variabili globali impiegate da GEOS 64, contengono anche informazioni utili per creare applicazioni che sfruttino alcune caratteristiche di GEOS 128.

GEOS V1.3 e le espansioni RAM

La versione 1.3 di GEOS, agli occhi dell'applicazione e dell'utente, fondamentalmente è solo un ampliamento della versione 1.2. Quindi tutte le routine della versione 1.2, i parametri, la struttura operativa e la dislocazione delle variabili globali, sono

mantenute integralmente nella versione 1.3. Ma alla struttura primitiva sono state aggiunte alcune altre capacità. Prima fra tutte, GEOS V1.3 è in grado di gestire le espansioni RAM REU (Ram Expansion Unit) e contiene alcune routine appositamente dedicate all'impiego della RAM aggiuntiva introdotta dai moduli d'espansione. Queste routine vengono illustrate nel capitolo 21 insieme alla compatibilità con GEOS 128. Le espansioni RAM sono strumenti di lavoro utilissimi, in quanto, riducendo notevolmente le necessità di accesso al disco, consentono di risparmiare tempo di lavoro e quindi di accelerare le operazioni. A seconda della quantità di memoria aggiuntiva, GEOS è in grado di impiegare il modulo d'espansione per muovere grandi quantità di dati in tempi brevissimi, per simulare un drive (RAM disk), per installare uno Shadowed Drive e per ricaricare velocemente il sistema senza compiere accessi al disco, per esempio dopo aver mandato in esecuzione un file Basic. Ma l'aspetto più notevole di questi possibili impieghi è che le applicazioni non sono tenute a sapere come e in che modo GEOS sta impiegando l'espansione RAM inserita, dal momento che la sua gestione nei casi sopracitati viene interamente affidata al Kernel. Comunque, le applicazioni possono anche impiegare le espansioni RAM per svolgere compiti del tutto diversi, chiamando le apposite routine messe a disposizione dal Kernel.

In GEOS V1.3 è stato migliorato ulteriormente il modulo che si interessa degli accessi al disco. In particolare è ora disponibile nella jump table a $C100 anche l'entry point della routine FreeBlock. Infine, nella nuova versione GEOS è in grado di gestire file di tipo AUTO_EXEC che vengono mandati in esecuzione automaticamente all'atto dell'installazione del sistema. Nel corso del manuale, quando verranno esposti argomenti che interessano esclusivamente la versione 1.3 di GEOS, sarà detto esplicitamente. In appendice vengono messe in evidenza tutte le costanti, le variabili e le routine che interessano esclusivamente la versione 1.3.

2 LE ICONE E I MENU

Le icone

Introducendo la gestione delle icone diamo per scontato che il lettore abbia già utilizzato GEOS e che quindi sappia cosa significa aprire un menu o selezionare un'icona. Non ci dilungheremo a illustrare che cosa sono e come agiscono. In breve, attivare o selezionare un'icona significa posizionare il puntatore del mouse nella zona di schermo dove è visibile l'icona e premere il pulsante: questa operazione determina la chiamata della routine di servizio associata all'icona. Il modo di operare è simile a quello adottato per i menu, se si eccettua il fatto che l'utente, selezionando le voci di un menu, può addentrarsi in successivi livelli di sotto-menu prima di attivare una voce che determini la chiamata della routine di servizio associata. La struttura dei menu consente quindi una gestione degli eventi più articolata e gerarchica di quanto non accada con la struttura delle icone. Le icone e i menu sono definiti tramite tavole di dati. Il modo più facile per imparare a servirsene è vedere praticamente come si creano: iniziamo quindi a illustrare l'argomento realizzando una tavola di dati che definisce alcune icone.

Di solito le icone vengono definite dalla routine d'inizializzazione dell'applicazione. Per esempio, deskTop definisce le icone associate ai file in maniera tale che effettuando una doppia pressione rapida del pulsante, quando il mouse si trova su una di esse, il file viene caricato in memoria dal Kernel di GEOS, il quale poi esegue un jsr all'indirizzo della routine d'inizializzazione dell'applicazione indicato nel blocco File Header. La semplice applicazione che ci apprestiamo a realizzare inizia all'indirizzo $0400. Il suo compito è solo quello di cancellare lo schermo chiamando i_Rectangle (routine che sarà descritta nel capitolo dedicato alla grafica), e successivamente definire le icone tramite la routine DoIcons.

Il programmatore deve creare la tavola di definizione delle icone, in maniera che poi il Kernel di GEOS vi possa accedere per ottenere tutte le informazioni di cui ha bisogno per gestire le icone desiderate. Per cedere al Kernel il controllo della tavola, l'applicazione deve passare alla routine DoIcons l'indirizzo di memoria dove questa è allocata.

```
LoadW       r0, QuantumTestIcons  ;indirizzo della tavola di dati delle icone
jsr         DoIcons               ;definisce e visualizza le icone
```

La tavola delle icone indica il numero di icone da definire e dove posizionare il mouse dopo la loro visualizzazione. La nostra semplice applicazione definisce otto icone. La tavola delle icone è composta da gruppi di sette byte per ogni icona.

Tavola delle icone

```
                                  ;La posizione x e y di un'icona e' riferita
                                  ;all'angolo superiore sinistro del suo disegno

X_POS_TOP_ICON = 3                ;coordinata x in byte dell'icona
Y_POS_TOP_ICON = 10               ;coordinata y in pixel dell'icona

QuantumTestIcons:
.byte       8                     ;numero di icone
.word       160                   ;coordinata x in pixel del mouse dopo
                                  ;l'apparizione delle icone
.byte       100                   ;coordinata y in pixel del mouse dopo
                                  ;l'apparizione delle icone

                                  ;GRUPPO 1
.word       showCaseData          ;puntatore ai dati grafici dell'icona
.byte       X_POS_TOP_ICON        ;coordinata x in byte dell'icona
.byte       Y_POS_TOP_ICON        ;coordinata y in pixel dell'icona
.byte       2, 16                 ;larghezza in byte e altezza in pixel dell'icona
.word       DoShow                ;routine di servizio associata all'icona showCase

                                  ;GRUPPO 2
.word       justForFunData        ;puntatore ai dati grafici dell'icona
.byte       X_POS_TOP_ICON        ;coordinata x in byte dell'icona
.byte       Y_POS_TOP_ICON+30     ;coordinata y in pixel dell'icona
.byte       2, 16                 ;larghezza in byte e altezza in pixel dell'icona
.word       DoFun                 ;routine di servizio associata all'icona justForFun

...                               ;altri 6 gruppi per le rimanenti 6 icone
```

Il gruppo di ogni singola icona inizia con un puntatore ai dati grafici necessari per disegnarla. I dati grafici per ogni icona nella nostra semplice applicazione sono composti da 33 byte. Le due successive informazioni contenute nel gruppo rappresentano la posizione che avrà sullo schermo (le coordinate di posizione x e y di qualsiasi icona sono sempre riferite all'angolo superiore sinistro del suo disegno). I due byte seguenti contengono rispettivamente la larghezza in byte e l'altezza in pixel dell'icona. Infine troviamo l'informazione più importante: la word che contiene l'indirizzo della routine di servizio associata all'icona. Nella nostra applicazione tutte le icone definite corrispondono, per semplicità, alla stessa routine di servizio.

Le due figure che seguono descrivono la routine di servizio della nostra applicazione e i dati grafici delle icone.

Routine di servizio

```
DoFun:                                ;routine di servizio per l'icona JustForFun
DoShow:                               ;routine di servizio per l'icona Commodore Software
ShowCase:
...                                   ;le rimanenti 6 label di servizio delle icone

jsr      i_Rectangle                  ;rettangolo che pulisce lo schermo
         .byte    0
         .byte    199
         .word    0
         .word    319

         LoadW    r0, Screen2Icon     ;indirizzo della tavola di dati per l'icona
         jsr      DoIcons             ;visualizza l'icona per ritornare a deskTop:
                                      ;dev'essere definita almeno un'icona,
                                      ;e' un bug di GEOS
         LoadW    r0,QuantumTestMenu  ;indirizzo della tavola di dati per il menu
         lda      #0                  ;posiziona il mouse sul primo menu
         jsr      DoMenu              ;visualizza il menu
rts
```

La routine di servizio appena descritta esegue tre compiti. Inizia cancellando lo schermo (disegnando un rettangolo di dimensioni uguali a quelle dello schermo con la matrice grafica di default, ovvero lo sfondo). La seconda operazione consiste nel definire una nuova struttura di icone che ne comprende solo una. La presenza di quest'icona è necessaria per due motivi: consente, se selezionata, di terminare l'applicazione e ricaricare deskTop, ma soprattutto è un'esigenza di GEOS. Si tratta di un piccolo bug del sistema operativo. GEOS richiede che sia sempre definita almeno un'icona, anche se non viene utilizzata. In seguito ne verranno spiegati più

dettagliatamente i motivi. La terza operazione che la routine di servizio esegue consiste nel definire e visualizzare la struttura di un menu. Alla fine, viene ceduto il controllo a MainLoop, che gestisce la nuova icona e il nuovo menu. Tutte le label delle routine di servizio della prima struttura di icone (QuantumTestIcons) che abbiamo descritto nell'esempio, individuano la routine qui illustrata.

Le routine di servizio delle icone possono eseguire qualunque operazione. Possono anche inizializzare nuovamente l'intero sistema e, se necessario, possono controllare quale tipo di pressione del pulsante del mouse, singola o doppia, è avvenuta sopra l'icona associata. Nel capitolo precedente avevamo spiegato come una routine associata a otherPressVec può gestire la doppia pressione del pulsante in aree non convenzionali, cioè diverse da icone e menu. Trattandosi di un caso un po' atipico, l'applicazione doveva procedere in modo autonomo all'analisi del contatore CLICK_COUNT e alle altre procedure necessarie.

Ora invece analizzeremo la doppia pressione del pulsante del mouse in un'area convenzionale: un'icona. In questo caso è il Kernel di GEOS che analizza il contatore in maniera completamente autonoma, sollevando l'applicazione da operazioni estranee ai suoi scopi. Gestire la doppia pressione del pulsante è molto più semplice quando il mouse si trova su un'icona, piuttosto che quando si trova su un'area non convenzionale (nel qual caso si sarebbe proceduto attraverso il vettore otherPressVec). La routine di servizio dell'icona selezionata, riceve in r0H, dal Kernel di GEOS, il valore TRUE ($FF) per indicare la doppia pressione, e FALSE ($00) per indicare la pressione singola. Il registro r0L viene restituito con il numero dell'icona. Questo numero corrisponde all'ordine di posizione lungo la tavola di definizione. La prima icona della tavola ha il numero 0. Conoscere il numero dell'icona attivata può apparire inutile se a ognuna è associata una routine particolare, che determina implicitamente quale selezione è avvenuta. Diventa importante, invece, se icone diverse hanno la stessa routine di servizio, ma questa viene eseguita in modo leggermente diverso in base al numero dell'icona selezionata.

Per concludere, dobbiamo definire i dati che comporranno il disegno in modo bit-map associato a ogni icona, e che saranno indicati in memoria dalla tavola di dati delle icone.

Tavola dei dati grafici

```
showCaseData:
  .byte        $02,$FF,$9C,$80,$01,$80,$39,$80,$6D,$80,$E5,$81,$BD,$83,$19,$86
  .byte        $31,$8C,$61,$98,$C1,$B1,$81,$BB,$01,$BE,$01,$BC,$01,$80,$01,$02
  .byte        $FF

justForFunData:
  .byte        $02,$FF,$9C,$87,$01,$8D,$81,$9E,$C1,$BB,$61,$AD,$B1,$B6,$99,$9B
  .byte        $09,$9D,$0D,$96,$07,$9A,$03,$8C,$01,$87,$E1,$8A,$31,$80,$19,$02
  .byte        $FF

  ...                               ;i dati grafici rimanenti per le altre 6 icone
```

Il disegno delle icone è in modo bit-map. Queste icone sono larghe 2 byte e alte 16 linee di scansione. I dati sono compattati nel modo BitmapUp (vedere BitmapUp nel capitolo dedicato alla grafica). Le mappe dei bit per queste icone non si compattano efficacemente: utilizzano infatti un byte in più del formato standard (la compattazione dei dati grafici può essere molto efficiente: per esempio l'intera riga di comandi in geoWrite occupa 16 byte). La lista completa dei dati grafici per ogni icona è riportata più avanti.

A questo punto della trattazione è importante introdurre la particolare tecnica di gestione delle icone utilizzata da deskTop e geoPaint. Nell'applicazione deskTop, quando l'utente preme il pulsante del mouse sopra un'icona associata a un file, l'icona si inverte per segnalare che è stata selezionata. L'icona del file rimane invertita fino a quando non viene selezionata un'altra icona. L'utente può poi aprire un menu e selezionare una voce: per esempio, info. La routine di servizio associata alla voce info preleva le informazioni dal file la cui icona è invertita. Una procedura simile è adottata anche da geoPaint quando viene selezionato un comando grafico. L'icona del comando rimane invertita fino a quando l'utente non ne attiva un'altra. Selezionando un'altra icona, la prima si inverte nuovamente per assumere lo stato normale di "non attivata", e la nuova icona si inverte per assumere lo stato di "attivata". Vediamo come avviene questa interazione, nel primo caso fra un'icona e un menu, e nel secondo caso fra due icone.

Nel Kernel di GEOS è presente una variabile denominata iconSetFlag. I bit 6 e 7 si occupano di come segnalare all'utente l'avvenuta selezione di un'icona. Se il bit 7 è impostato a 1, l'icona viene solo momentaneamente invertita per indicare che è stata selezionata. Il programmatore, per impostare a 1 questo bit, dispone della costante SET_FLASH = $80, che consente di non dover ricordare a memoria quale bit e quale valore impostano questa opzione. Il tempo durante il quale l'icona rimane invertita

è determinato dalla variabile selectionFlash. Il suo valore di default è SELECTION_DELAY = 10.

Se il bit 7 è impostato a 0 e il bit 6 a 1, l'icona selezionata si inverte e rimane in questo stato. La costante per questa opzione è ST_INVERT = $40. In questo caso la routine di servizio associata all'icona deve memorizzare nelle variabili dell'applicazione lo stato dell'icona, il suo numero e qualsiasi altra informazione si renda necessaria. Quando poi viene selezionata la voce di un menu, la routine di servizio associata a questa voce esamina le variabili dell'applicazione per scoprire se c'è un'icona in stato di "attivata", cioè invertita, e operare le appropriate scelte. Nell'esempio della voce info, la routine di servizio dell'icona appena selezionata inverte nuovamente l'icona tramite la routine InvertRectangle, riportandola così allo stato di non attivata, e preleva le informazioni dal file per mostrarle all'utente.

Se entrambi i bit 6 e 7 di iconSetFlag sono azzerati, l'attivazione dell'icona non produce alcun cambiamento nel suo disegno.

Una nota importante prima di terminare la trattazione sulle icone: GEOS, per come è stato realizzato, esige che l'applicazione definisca sempre almeno un'icona. Se l'applicazione che state creando non deve utilizzare alcuna icona, è necessario crearne una ugualmente. Definitela larga un byte e alta una linea di scansione, e impostate il suo puntatore ai dati grafici del disegno uguale a 0. In questo modo non siete costretti a definire un disegno fittizio e l'icona non potrà mai essere selezionata.

Il Kernel di GEOS prevede una sola routine per attivare le icone: DoIcons.

DoIcons

Funzione: Disegna e attiva le icone definite nella Icon Table (tavola di definizione delle icone).

Indirizzo: $C15A

Parametri: r0 puntatore alla Icon Table

Restituisce: Niente

Distrugge: a, x, y, r0 - r11

Sinossi: DoIcons disegna le icone come sono definite nella Icon Table puntata da r0. Quando sono state disegnate, l'utente può selezionarle tramite il mouse. Attivando un'icona, viene eseguita la routine di servizio indicata nella Icon Table.

Per disattivare completamente un'icona, bisogna azzerare il suo puntatore ai dati grafici. Se l'icona è già stata visualizzata bisognerà cancellarne il disegno dallo schermo. Se il puntatore ai dati grafici è azzerato, non apparirà alcun disegno.

L'unico modo affidabile per disattivare un intero set di icone visualizzate è cancellare lo schermo e inizializzarne un altro, oppure azzerare tutti i puntatori ai dati grafici.

Il Kernel di GEOS comunica alla routine di servizio associata all'icona, tramite il registro r0H, il valore TRUE ($FF) per indicare la doppia pressione del pulsante del mouse sull'icona, e il valore FALSE ($00) per indicare la pressione singola. Il Kernel comunica inoltre, nel registro r0L, il numero dell'icona corrispondente al numero d'ordine con il quale appare nella Icon Table. La prima icona è numerata con 0. Questo numero d'ordine è utile soprattutto quando accade che a icone diverse sia associata la stessa routine di servizio, senza però che il *comportamento* di questa routine sia lo stesso per ogni icona selezionata.

I menu

Come per le icone, procediamo a illustrare il funzionamento dei menu con un esempio. La prima cosa da fare è realizzare una chiamata come la seguente:

```
LoadW          r0, TestMenu        ;puntatore alla tavola di definizione del menu
jsr            DoMenu              ;definisce il menu
```

Esattamente come per le icone, i menu sono definiti attraverso la tavola dei dati. La semplice applicazione che stiamo creando contiene la struttura completa di un menu, che analizzeremo un passo alla volta.

I menu possono aprirsi sia verticalmente sia orizzontalmente. In deskTop la linea comandi che compare sullo schermo in alto a sinistra è un menu orizzontale. I primi quattro parametri che compongono la tavola di dati sono le dimensioni geometriche del menu: dove iniziano i lati superiore, inferiore, destro e sinistro. Il lato superiore coincide di solito con il bordo superiore dello schermo, e il lato inferiore si calcola tenendo conto che ogni voce, nel caso di un menu verticale, occupa 14 pixel (dimensione ottimale quando si utilizza il set di caratteri standard). Quindi al lato superiore aggiungiamo 14 pixel per ogni voce, in modo da ottenere la posizione del lato inferiore del menu.

Il lato destro di un menu orizzontale crea qualche problema in più. Innanzitutto i menu orizzontali si aprono verso destra. Questo significa che se apriamo un menu verticale e selezioniamo una voce che apre un altro menu orizzontale (struttura nidificata dei menu), quest'ultimo si apre a destra della voce. Il problema della determinazione del lato destro è che la larghezza di una voce dipende dalla lunghezza della parola che contiene. Dal momento che i caratteri non sono tutti delle stesse dimensioni e la spaziatura è proporzionale, non è a priori possibile per il programmatore stabilire l'esatta larghezza del menu orizzontale. Tale dimensione si determina sperimentalmente, iniziando con un tentativo sovradimensionato e controllando che il box dell'ultima voce a destra si inverta completamente e non parzialmente. Nel corso di questa operazione, bisogna anche tener presente la possibilità che il menu sia troppo corto e che selezionando l'ultima voce a destra venga non solo invertito il box, ma anche parte dello sfondo. Sono quindi necessari alcuni tentativi prima di trovare la dimensione appropriata. Fate attenzione, però, perché possono verificarsi strani e imprevedibili bug se il box della voce ha dimensioni tali da invertire parti dello sfondo.

Basandosi sulle dimensioni definite nella tavola di dati, GEOS disegna un lungo box per creare il menu. Successivamente scrive le parole chiave delle singole voci una dopo l'altra sulla stessa linea, separandole fra di loro con una barretta verticale |. Le barrette verticali dividono il menu in voci separate. Se il valore impostato per la posizione del lato destro del menu orizzontale è sovradimensionato, l'ultima voce si limiterà ad avere

molti spazi dopo la parola. In questo caso è semplice procedere alla regolazione ottimale della larghezza del menu.

La quinta informazione definisce il tipo di menu, orizzontale o verticale, e il numero di voci di cui è composto. Questi due parametri sono contenuti nello stesso byte perché occupano bit diversi; l'informazione è costituita dal risultato dell'operazione OR fra i due valori. I primi cinque parametri dovrebbero apparire così:

Dimensioni del menu

```
                              ;Costanti per le dimensioni del menu
 MAIN_TOP      =    10        ;lato superiore del menu
 MAIN_BOT      =    24        ;lato inferiore del menu
 MAIN_LFT      =    0         ;lato sinistro del menu
 MAIN_RT       =    255       ;lato destro (sperimentale) del menu

QuantumTestMenu:

 .byte         MAIN_TOP       ;lato superiore del menu
 .byte         MAIN_BOT       ;lato inferiore del menu
 .word         MAIN_LFT       ;lato sinistro del menu
 .word         MAIN_RT        ;lato destro del menu
 .byte         HORIZONTAL|2   ;tipo del menu OR'ed(†) con il numero delle voci del menu
```

La selezione dei menu

Passiamo a descrivere in dettaglio i parametri associati a ogni voce che appare nel menu. Ogni voce è caratterizzata da tre parametri. Il primo è un puntatore alla stringa di testo che appare nella voce. La stringa di testo può contenere qualsiasi carattere visualizzabile, compresi gli spazi fra le parole. La seconda informazione è un byte che indica il tipo di voce del menu. La selezione di un menu può aprire un sotto-menu o causare la chiamata di una routine di servizio. In deskTop le voci del menu principale orizzontale aprono altri sotto-menu. A loro volta le voci dei sotto-menu possono aprire altri sotto-menu o chiamare routine di servizio. La terza informazione è un puntatore che, a seconda del caso, contiene l'indirizzo della tavola di dati associata al

(†) L'operazione logica OR fra due valori è indicata per brevità con il verbo OR'ed. Per esempio, HORIZONTAL OR'ed con 7 indica che il risultato è costituito dall'operazione logica OR eseguita fra il valore della costante HORIZONTAL e il numero 7.

sotto-menu, o l'indirizzo della routine di servizio da eseguire. Le due voci del menu orizzontale della nostra applicazione d'esempio sono illustrate nella seguente figura.

Voci del menu

```
.word    ChangeDeptText    ;puntatore alla stringa di testo terminante
                           ;con uno 0 (stringa a terminazione nulla)
.byte    SUB_MENU          ;flag per indicare che questa voce apre un sotto-menu
.word    ChangeDeptMenu    ;puntatore alla struttura del sotto-menu

.word    ShowCaseText      ;puntatore alla stringa a terminazione nulla
.byte    SUB_MENU          ;flag per indicare che questa voce apre un sotto-menu
.word    ShowCaseMenu      ;puntatore alla struttura del sotto-menu
```

Per completezza, anche se l'applicazione d'esempio che stiamo analizzando sarà interamente ripresa per essere realizzata in pratica, seguono le stringhe di testo a cui fa riferimento la tavola della figura precedente:

```
ChangeDeptText: .byte    "Change Departments", 0
ShowCaseText:   .byte    "Commodore Software Showcase", 0
```

Le strutture dei due sotto-menu richiamati nella precedente figura sono illustrate nella pagina successiva. L'esempio completo apparirà nel capitolo seguente. Dal momento che le strutture dei menu sono ripetitive, ne analizzeremo uno o due esempi per ogni tipo, rimandando il lettore al prossimo capitolo per la trattazione completa.

La struttura di un sotto-menu è simile a quella dei menu principali. Le prime scelte da operare riguardano la posizione che deve assumere il sotto-menu quando viene visualizzato. Nella nostra applicazione, la voce Change Departments del menu principale controlla un sotto-menu verticale che si estende sotto di essa. In questo caso la dimensione verticale del sotto-menu viene calcolata tenendo conto del numero di voci di cui è composto e allineandolo esattamente sotto il menu principale. Per quanto riguarda le dimensioni orizzontali, è ancora necessario procedere per tentativi. Si deve allineare il lato sinistro del sotto-menu con la barretta verticale che corrisponde all'inizio della voce del menu principale che lo controlla, mentre il lato destro deve consentire alla voce più larga del sotto-menu di invertirsi correttamente. Ovviamente, trattandosi di un menu verticale, le voci più corte hanno spazio in eccesso alla loro destra, ma il Kernel di GEOS inverte i loro box fino al lato verticale destro. La determinazione delle dimensioni del menu verticale sarà più chiara rifacendosi a un esempio. I primi parametri del sotto-menu ChangeDeptMenu, le dimensioni e il tipo,

mostrano che è un menu verticale. È composto da otto voci e ognuna attiva un MENU_ACTION, cioè una routine di servizio, anziché altri sotto-menu(†). Nella figura che segue appaiono solo le definizioni delle prime due voci, dal momento che la struttura non cambia ed è la stessa che abbiamo analizzato per il menu principale.

Le voci del sotto-menu Change Departments

```
ChangeDeptMenu:
  .byte    MAIN_BOT           ;il lato superiore del sotto-menu coincide con il lato
                              ;inferiore del menu principale
  .byte    MAIN_BOT+8*14+1    ;il lato inferiore del sotto-menu viene determinato
                              ;aggiungendo 14 linee di scansione per voce
  .word    MAIN_LFT           ;lato verticale sinistro del sotto-menu
  .word    MAIN_LFT+155       ;lato verticale destro del sotto-menu
  .byte    VERTICAL|8         ;tipo menu OR'ed con il numero di voci

  .word    ShowCaseText       ;testo per questa voce
  .byte    MENU_ACTION        ;flag per indicare una routine di servizio
  .word    ShowCaseDsp        ;indirizzo della routine di servizio

  .word    FunText            ;testo per questa voce
  .byte    MENU_ACTION        ;flag per indicare una routine di servizio
  .word    FunDsp             ;indirizzo della routine di servizio

  ...                         ;altri 6 gruppi di dati per le voci rimanenti
```

Questo sotto-menu è composto da otto voci, due delle quali sono riportate nella figura precedente. Le rimanenti sei sono dello stesso tipo. Il lato superiore del sotto-menu di solito è posizionato in corrispondenza del lato inferiore del menu principale che l'ha generato (non rappresenta una regola fissa). Ogni voce del sotto-menu verticale misura in altezza 14 linee di scansione. GEOS aggiungerà una linea di scansione in più al lato inferiore. Il lato inferiore del menu può essere determinato come segue:

```
Lato inferiore = Lato superiore + (Numero di voci * 14) + 1
```

(†) È possibile creare una struttura di sotto-menu orizzontali e verticali con una profondità arbitraria. La selezione della voce di un menu orizzontale può aprire un sotto-menu verticale, le cui voci a loro volta possono aprire altri sotto-menu orizzontali e così via, generando una struttura di menu a elevata profondità.

Questa è l'unica dimensione dei menu che è possibile determinare con esattezza senza dover ricorrere a una serie di tentativi. Questo sotto-menu è definito con il lato verticale sinistro nella stessa posizione del lato sinistro del menu principale. La posizione del lato destro dev'essere determinata sperimentalmente. La prima voce contiene il testo puntato da ShowCaseText, e quando è attivata causa un MENU_ACTION: viene eseguita la routine di servizio ShowCaseDsp, puntata dalla word successiva nel gruppo di dati.

Le altre voci sono del tutto simili. Dopo la struttura del sotto-menu ChangeDeptMenu, segue quella associata a ShowCaseMenu. Questo secondo sotto-menu, controllato dalla seconda voce del menu principale, è uguale al primo che abbiamo trattato, a parte il fatto che è un sotto-menu obbligato (constrained). Il gruppo di dati per questo sotto-menu è riportato nel listato dell'applicazione esemplificativa che si trova (completo) nel prossimo capitolo. "Obbligato" significa che il mouse non può uscire dai suoi bordi. Questa opzione è utile per permettere all'utente di muoversi velocemente all'interno del sotto-menu aperto senza correre il rischio che il mouse esca dai limiti del sotto-menu e attivi la scomparsa dello stesso. È una caratteristica molto utile per menu con diverse voci, o menu orizzontali aperti da un menu verticale, dove uno spostamento accidentale del mouse può chiudere entrambi i menu. L'indicatore dello stato d'obbligo è un bit nel byte del tipo di menu:

```
.byte       CONSTRAINED|VERTICAL|7    ;menu obbligato, tipo di menu, numero di voci
```

Dopo ShowCaseMenu sono indicate 15 stringhe di testo per le voci dei due sotto-menu, anche queste riportate nel prossimo capitolo. Per finire rimangono da definire le routine di servizio associate alle voci dei sotto-menu. Queste routine possono eseguire qualunque operazione, ma devono includere almeno una chiamata a una routine di gestione dei menu, come GotoFirstMenu, per esempio, che disattiva tutti i sotto-menu aperti, facendoli scomparire e lasciando soltanto il menu principale. Le nostre routine di servizio eseguono solo questa operazione, chiamando GotoFirstMenu.

```
LearningDsp:            ;routine di servizio del sotto-menu associato alla voce
FunDsp:                 ;Change Departments del menu principale
ServiceDsp:
InfoNetDsp:
PeopleConDsp:
NewsDsp:
MallDsp:

CatalogDsp:             ;routine di servizio del sotto-menu associato alla voce Software
PreviewDsp:             ;ShowCase
SigLibDsp:
```

```
FileTransDsp:
ReviewDsp:
PostOfficeDsp:
ChangeDsp:
ShowCaseDsp:

    jsr        GotoFirstMenu        ;disattiva tutti i sotto-menu
    rts
```

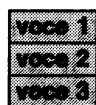
Questo è tutto quello che c'è da sapere per costruire una semplice struttura di menu. GEOS permette di effettuare molte variazioni alla struttura principale. Per esempio, prevede una routine che può essere impiegata al posto di GotoFirstMenu, DoPreviousMenu, la quale chiude solo l'ultimo sotto-menu aperto. Questa routine permette all'utente di selezionare più voci dello stesso menu senza doverlo riaprire. Per esempio, supponiamo di avere un menu verticale con le voci A, B, C, e D. Ognuna di queste voci apre un menu orizzontale con le voci 1, 2, 3, e 4. L'utente desidera selezionare l'opzione 1 del sotto-menu corrispondente alla voce D, e l'opzione 3 del sotto-menu corrispondente alla voce C. Quando l'utente seleziona l'opzione 1 sotto D, la routine di servizio associata chiama DoPreviousMenu, e anziché produrre la scomparsa di tutti i sotto-menu, cancella solo il sotto-menu orizzontale D. In questo caso la routine di servizio deve anche preoccuparsi di aggiornare le variabili mouseXPos e mouseYPos in modo che il mouse, dopo la scomparsa del sotto-menu, si trovi sul menu verticale. In caso contrario, quando il Kernel di GEOS chiude il sotto-menu orizzontale, il mouse non si trova sul menu verticale e il Kernel interpreta questa situazione come l'uscita dell'utente con il mouse dal menu verticale, e di conseguenza chiude anche il menu verticale.

Le routine di servizio associate ai menu possono essere anche molto sofisticate. Possono perfino modificare se stesse o la struttura dei menu (un'operazione comunque sconsigliabile per chi è alle prime armi). Per esempio, la routine di servizio può aggiungere un asterisco al testo di una voce di modo che quando il sotto-menu viene aperto per la seconda volta, l'utente può sapere quali voci ha già attivato. Questo accorgimento viene utilizzato in geoWrite per mostrare se opzioni come bold (nero) o italic (corsivo) sono già state utilizzate. Ogni volta che una di queste opzioni viene selezionata, la routine di servizio esegue i suoi compiti primari, e inoltre accede alla tavola di dati associata al menu e aggiunge un asterisco (o lo cancella se è già presente) al testo della voce selezionata.

C'è un altro tipo di menu chiamato DYN_SUB_MENU che permette al programmatore di creare una routine da eseguire prima che il menu venga veramente aperto sullo schermo. La word che segue .byte DYN_SUB_MENU è un puntatore che contiene l'indirizzo della routine da eseguire prima che venga aperto il sotto-menu. Per i menu normali, questa word punta alla struttura del sotto-menu; quindi la routine di preparazione, eseguita prima dell'apertura del sotto-menu, deve aggiornare r0 con

l'indirizzo della struttura del sotto-menu da aprire, e infine terminare con un solo rts.
I sotto-menu dinamici sono utili per creare una struttura di menu elastica, cioè adattabile a situazioni particolari. La routine può controllare lo stato del sistema e modificare la tavola di dati associata, prima che il sotto-menu venga aperto. Nel capitolo dedicato alla gestione dei testi è illustrato un metodo per creare il menu dei corpi carattere quando si sceglie una fonte carattere (come appare in geoWrite e geoPaint). All'atto della scelta della fonte carattere, la routine di preparazione dinamica del sotto-menu controlla quali corpi carattere sono associati alla fonte prescelta e costruisce la tavola di dati per il sotto-menu. Le varie possibilità di gestione dei menu, inclusi i menu dinamici, sono riassunte nello schema che segue.

Le costanti per i menu

Tipo di menu

HORIZONTAL	Le voci del menu sono disposte orizzontalmente voce 1 | voce 2 | voce 3
VERTICAL	Le voci del menu sono disposte verticalmente voce 1 voce 2 voce 3
CONSTRAINED	Il mouse e' obbligato a non uscire dai lati e dal bordo inferiore del menu. Se non si effettuano selezioni, si puo' chiudere il menu spostando il mouse oltre il lato superiore

Tipo di selezione del menu

SUB_MENU	Questa scelta apre un altro livello di sotto-menu. E' seguita da un puntatore alla struttura del sotto-menu
MENU_ACTION	Questa scelta determina l'esecuzione della routine di servizio associata, il cui indirizzo e' contenuto nel puntatore che segue
DYN_SUB_MENU	Questa scelta determina la chiamata a una routine di preparazione, prima che il sotto-menu venga aperto. La routine, quando termina, deve impostare r0 con l'indirizzo della struttura del sotto-menu

DoMenu

Funzione: Disegna e attiva la struttura dei menu puntata da r0.

Indirizzo: $C151

Parametri:

a	numero dalla voce su cui si posiziona il mouse
r0	indirizzo della tavola di dati

Restituisce: Niente

Distrugge: r0 - r13, a, x, y

Sinossi: Cedendo il controllo della tavola di definizione, DoMenu disegna i menu e ne cede il controllo al Kernel. Se la tavola di definizione è corretta, DoMenu è in grado di gestire una struttura di menu di livello arbitrario. Dopo che i menu sono stati inizializzati da DoMenu, la loro manipolazione viene eseguita dal Kernel di GEOS in MainLoop. Premendo il pulsante del mouse sopra la voce di un menu, il Kernel di GEOS apre un sotto-menu o chiama la routine di servizio.

DoMenu è normalmente chiamata dalla routine d'inizializzazione dell'applicazione (GEOS dopo aver caricato un'applicazione la manda in esecuzione all'indirizzo d'inizializzazione specificato nel File Header, indirizzo al quale si trova in genere anche una chiamata alla routine DoMenu).

ReDoMenu

Funzione: Riabilita un menu, cioè fa in modo che non scompaia quando una voce viene selezionata, in modo da permettere all'utente di selezionarne un'altra.

Indirizzo: $C193

Parametri: Nessuno

Restituisce: Niente

Distrugge: r0 - r13, a, x, y

Sinossi: Il codice di MainLoop che gestisce la struttura dei menu è a conoscenza dei dati relativi all'ultimo livello di menu aperto. Quando viene attivata una voce di tipo MENU_ACTION , MainLoop chiama la routine di servizio associata. All'interno di questa routine il programmatore deve indicare a GEOS come utilizzare l'ultimo livello di menu aperto. ReDoMenu riabilita l'ultimo livello di menu in modo che si possa selezionare un'altra voce senza dover riaprire il sotto-menu.

DoPreviousMenu

Funzione: Disabilita il sotto-menu correntemente aperto e lascia abilitati (cioè non chiude) gli altri menu eventualmente aperti.

Indirizzo: $C190

Parametri: Nessuno

Restituisce: Niente

Distrugge: r0 - r13, a, x, y

Sinossi: Il codice di MainLoop che gestisce la struttura di menu è a conoscenza dei dati relativi all'ultimo livello di menu aperto. Quando viene attivata una voce di tipo MENU_ACTION, MainLoop chiama la routine di servizio associata. All'interno di questa routine il programmatore deve indicare a GEOS come utilizzare l'ultimo livello di menu aperto.

DoPreviousMenu chiude l'ultimo menu, e lascia abilitata tutta la precedente struttura già aperta. Il mouse è abilitato in modo da permettere all'utente di selezionare un'altra voce del menu precedente. Prima di chiamare DoPreviousMenu, bisogna ricollocare il mouse sul menu precedentemente aperto, altrimenti il Kernel di GEOS interpreterà la situazione corrente come segue:

1) si accorge che il mouse si trova fuori dai limiti del menu
2) presume che l'utente abbia spostato il mouse al di fuori del menu per cancellarlo
3) cancella tutti i sotto-menu aperti.

GotoFirstMenu

Funzione: Chiude tutti i sotto-menu aperti. Viene chiamata dalla routine di servizio della voce.

Indirizzo: $C1BD

Parametri: Nessuno

Restituisce: Niente

Distrugge: r0 - r13, a, x, y

Sinossi: Quando viene attivata una selezione di tipo MENU_ACTION, il codice di gestione dei menu contenuto in MainLoop chiama la routine di servizio associata alla voce. La routine deve indicare a GEOS come utilizzare la struttura di menu aperta.

GotoFirstMenu disabilita tutta la struttura dei sotto-menu correntemente aperta e lascia attivo solo il menu principale.

RecoverMenu

Funzione: Ripristina lo sfondo coperto dall'ultimo menu copiandolo dal buffer di schermo. La routine non riattiva il livello superiore di menu.

Indirizzo: $C154

Parametri: Nessuno

Restituisce: Niente

Distrugge: r2 - r8, r11, a, x, y

Sinossi: Questa routine ripristina lo sfondo coperto dal menu corrente. A differenza di DoPreviousMenu, RecoverMenu non riattiva il menu di livello precedente.

RecoverAllMenus

Funzione: Ripristina lo sfondo coperto da tutti i menu aperti, compreso il menu principale.

Indirizzo: $C157

Parametri: Nessuno

Restituisce: Niente

Distrugge: r2 - r8, r11, a, x, y

Sinossi: Ripristina lo sfondo coperto da tutti i menu correntemente aperti, compreso il menu principale. A differenza di GotoFirstMenu, RecoverAllMenus disattiva completamente la struttura dei menu cancellandola interamente dallo schermo.

Realizzazione ed esecuzione dell'applicazione

Abbiamo trattato gli argomenti principali per impiegare i menu e le icone e siamo ora in grado di creare una semplice applicazione che li utilizzi. Il passo successivo consiste nel realizzarla praticamente ed eseguirla. Prima di continuare dobbiamo preparare l'ambiente di lavoro e rendere accessibili all'applicazione le informazioni simboliche di GEOS: indirizzi delle routine, costanti, variabili.

Per i programmatori che desiderano realizzare applicazioni in ambiente GEOS, la Berkeley Softworks ha prodotto geoProgrammer. Si tratta di un pacchetto professionale pensato espressamente per creare applicazioni GEOS compatibili (se necessario può essere usato anche per applicazioni non GEOS compatibili).

Il pacchetto raccoglie tre strumenti fondamentali per il programmatore: un assembler, un linker e un debugger. geoAssembler, oltre a leggere i file sorgente creati con geoWrite, gestisce numerosi pseudocodici per creare qualsiasi tipo di compilazione condizionale. Dal momento che il file sorgente viene creato con geoWrite, può contenere al suo interno qualsiasi elemento atto a migliorare la presentazione grafica del documento e la sua leggibilità. Nel testo possono essere incluse, ad esempio, anche figure, che geoAssembler provvede a trasformare in codici automaticamente. geoLinker assembla i vari moduli di un'applicazione in uno stesso file eseguibile. Tramite geoLinker, il programmatore può creare file con struttura SEQUENTIAL, VLIR e file Desk Accessory (accessori da scrivania). geoDebugger permette di collaudare l'applicazione direttamente in memoria disassemblando i codici delle istruzioni e permettendo all'utente di correggerli. Questa applicazione consente anche di mandare in esecuzione un'istruzione alla volta e di eseguire le subroutine come singole istruzioni. Inoltre permette di interrompere l'esecuzione dell'applicazione con la pressione di un tasto e di stabilire fino a 8 breakpoint condizionali. Quando il programma incontra un breakpoint, geoDebugger apre una finestra indipendente dall'applicazione sottoposta al test, nella quale visualizza l'errore. Nel pacchetto geoProgrammer, oltre a questi fondamentali strumenti di lavoro, il programmatore ha a disposizione anche una serie di applicazioni GEOS compatibili incluse come esempi pratici.

Se non si dispone del pacchetto geoProgrammer, si deve affrontare un problema di non semplice soluzione: i file utilizzati da GEOS sono diversi dai file standard gestiti dal C-64. I compilatori Assembly correntemente in commercio non sono in grado di salvare i file in standard GEOS. Dobbiamo quindi trovare un modo per scavalcare il problema.

A questo scopo si utilizza uno stratagemma che consente di realizzare l'applicazione con un file PRG standard C-64. Dopo aver creato l'applicazione nel formato PRG, si manda in esecuzione un semplice programma in Basic che traduce il file PRG in un file GEOS di tipo APPLICATION. Trattiamo questo argomento nei dettagli, prima di proseguire alla realizzazione dell'applicazione, per essere padroni degli strumenti necessari a un lavoro quanto più possibile rapido ed efficiente.

Lo scopo di questo capitolo è risolvere tutti i problemi in cui il programmatore può incappare realizzando ed eseguendo nel proprio ambiente di lavoro un'applicazione in standard GEOS.

Le costanti, le variabili, le routine e le macro

La semplice applicazione che troverete nel prossimo capitolo è stata prodotta impiegando il nostro compilatore Assembly. Il vantaggio d'impiegare un linguaggio simbolico nei file sorgente dell'applicazione sta nella migliore leggibilità dei codici che sono attualmente in esecuzione nel C-64. Il file sorgente dell'applicazione inizia con una serie di linee contenenti la direttiva ".include nomefile". La direttiva ".include" indica al nostro compilatore che in quel punto del file sorgente dev'essere inserito un testo proveniente da un altro file il cui nome è indicato da nomefile. Per poter utilizzare tutti gli pseudonimi per le costanti, le variabili, le routine e le macro, ogni applicazione deve contenere nel suo file sorgente la direttiva .include per richiamare i file geosConstants, geosMemoryMap, geosRoutines e geosMacros. "geosConstants" contiene la definizione di tutte le costanti di sistema utili alla programmazione. "geosMemoryMap" contiene la definizione di tutte le variabili e dei buffer di sistema. "geosRoutines" contiene gli indirizzi associati agli pseudonomi delle routine di sistema; questi indirizzi sono contenuti nella jump table del sistema. "geosMacros" contiene tutte le definizioni delle macro istruzioni impiegate in questo libro.

Questi file sono contenuti in appendice. Il primo lavoro che il programmatore dovrebbe svolgere è quello di includerli nel proprio sistema, in modo da renderli disponibili per ogni applicazione. Eseguita questa operazione, è buona norma preparare alcune copie di sicurezza. Se il vostro compilatore Assembly è in grado di gestire solo un file alla volta, cioè non è in grado di inserire i testi di file esterni durante la compilazione, allora dovete inserirli in ogni file che create per realizzare le applicazioni. Può inoltre accadere che il vostro compilatore non accetti label di lunghezza superiore a 8 o 9 caratteri.

Quando questo noioso lavoro è stato svolto, il vostro ambiente di lavoro è pronto per iniziare a compilare i file sorgente delle applicazioni che creerete. Una buona idea per iniziare a programmare consiste nel realizzare la semplice applicazione che vi proponiamo, o una simile, in modo da operare i ritocchi necessari al vostro sistema di lavoro perché si adatti perfettamente alla struttura di programmazione richiesta da GEOS. Nel pacchetto di programmazione geoProgrammer i file geosConstants, geosMemoryMap, geosRoutines e geosMacros sono già disponibili.

La struttura dell'applicazione

I file in formato GEOS contengono un blocco extra, rispetto ai file normali, chiamato blocco File Header. Questo contiene i dati grafici dell'icona associata al file e altre informazioni necessarie a GEOS per documentare il file e manipolarlo correttamente,

come l'indirizzo al quale allocarlo in memoria e l'indirizzo al quale mandarlo in esecuzione.

L'indirizzo al quale allocarlo indica da che punto in poi nella memoria del calcolatore dev'essere caricato il file. L'indirizzo al quale eseguirlo indica a GEOS dove si trova la routine d'inizializzazione dell'applicazione che dev'essere eseguita quando il file viene caricato in memoria.

Poiché, nel momento in cui scriviamo, geoProgrammer è l'unico pacchetto di programmazione GEOS compatibile presente in commercio, analizziamo un modo alternativo di creare applicazioni GEOS compatibili con qualsiasi compilatore. Il guaio è che i compilatori in commercio non sono in grado di creare il blocco File Header. Si possono utilizzare due stratagemmi per scavalcare il problema. Il primo consiste nell'utilizzare le routine di GEOS per salvare il file dalla memoria al disco. Sfortunatamente però, dal momento che GEOS inizializza la memoria a uno stato di default prima di eseguire l'applicazione, il programmatore dovrebbe caricare GEOS e poi trovare il modo di interrompere la sua esecuzione. Se successivamente riesce a trovare il modo di caricare l'applicazione in memoria, potrebbe includere nei suoi codici alcune chiamate alle routine del Kernel di GEOS per salvare l'applicazione su disco con il nuovo standard GEOS. Per farlo si può ricorrere a speciali unità hardware chiamate unità ICE (In-Circuit-Emulator).

Fortunatamente possiamo trasformare il file PRG dell'applicazione nello standard GEOS in un altro modo. Questo prevede che il vostro compilatore salvi il file oggetto in memoria, nel formato normale PRG del C-64. Il File Header viene incluso come primo blocco del file oggetto, e quindi nel file sorgente si trova all'inizio, di modo che, quando è compilato, occupa esattamente il primo settore del file. Un programma Basic si preoccupa poi di prelevare il File Header, allocarlo su disco separatamente e riagganciare il resto del programma. Vediamo quali sono i dati contenuti nel blocco File Header e qual è il loro significato.

Il blocco File Header

Lo schema che segue nella pagina successiva mostra la struttura di un file in standard GEOS. Come normalmente accade, il File Entry in directory punta al primo blocco di dati del file. Rispetto al formato dei normali File Entry, il File Entry di un file GEOS compatibile contiene anche il puntatore al blocco File Header. In un normale file standard C-64, questi byte puntano al primo side-sector di un file relativo. Dal momento che GEOS non utilizza i file relativi, questi due byte sono disponibili per puntare al blocco File Header.

Struttura di base dei file GEOS compatibili

File Entry del file

I byte 19 - 20 indicano l'indirizzo T/S del blocco File Header

Blocco File Header

I byte 1 - 2 indicano l'indirizzo T/S del primo blocco del file

Primo blocco del file

Byte 0 - 1

T/S blocco successivo

Secondo blocco del file

Byte 0 - 1

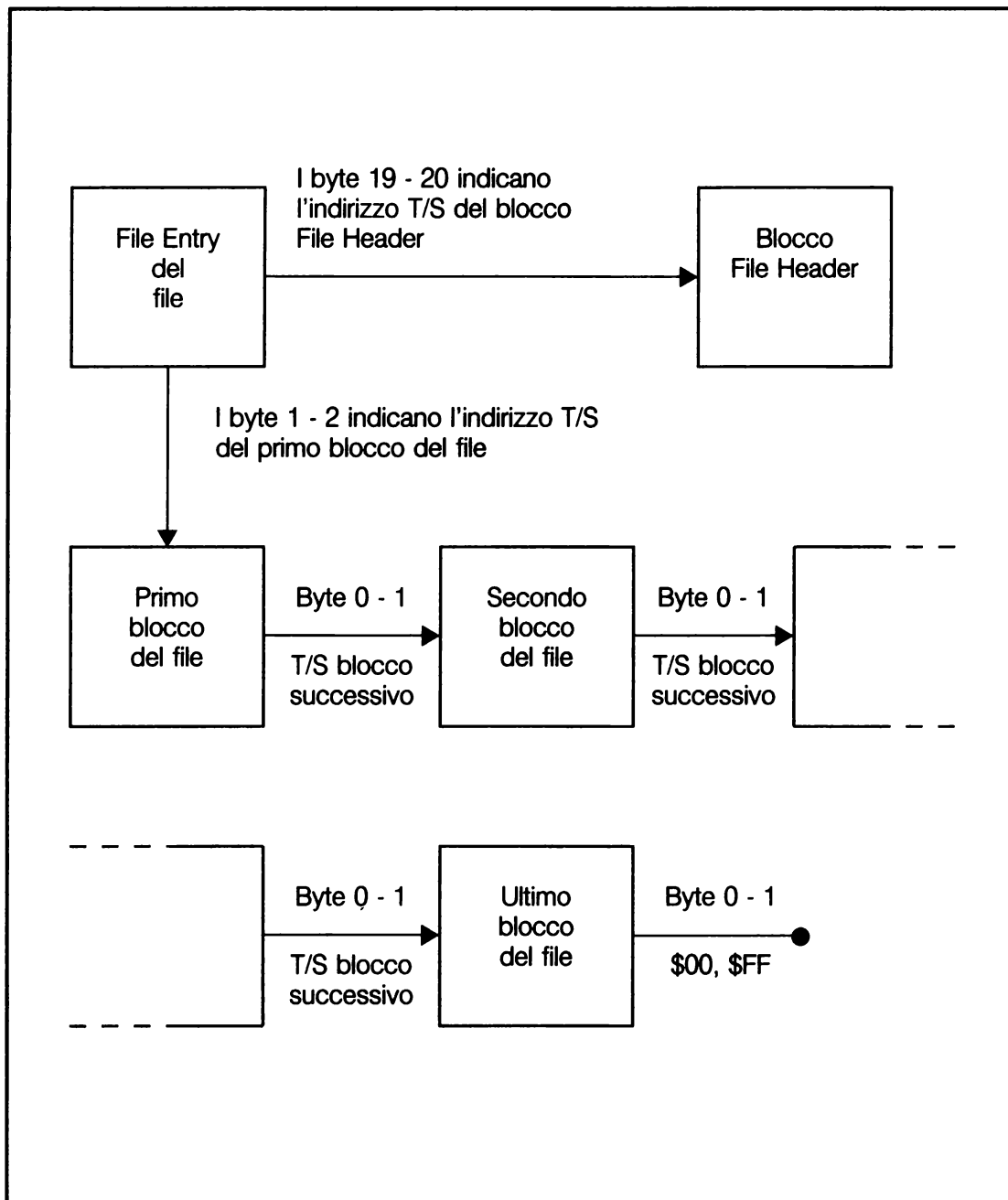
T/S blocco successivo

Byte 0 - 1

T/S blocco successivo

Ultimo blocco del file

Byte 0 - 1

$00, $FF

Il trucco che utilizziamo per permettere al compilatore di salvare un file su disco in formato GEOS consiste nell'includere il blocco File Header come primo blocco del file. Dal momento che la memoria disponibile per le applicazioni in formato GEOS inizia all'indirizzo $0400, conviene scrivere il file sorgente dell'applicazione, compreso il File Header, in modo che il compilatore inizi a compilare i codici dell'applicazione a $0400. Dal momento che il primo blocco di un file su disco contiene l'indirizzo traccia e settore (T/S) del blocco successivo nei primi due byte, e nei due successivi l'indirizzo al quale allocarlo in memoria, sono disponibili solo 252 byte per il File Header, che deve occupare esattamente, le dimensioni di un blocco e non superarle. Quindi il compilatore deve iniziare a creare il file oggetto, comprendente il blocco File Header, a $0304 = $0400 – 252. In questo modo, quando il file è stato compilato e salvato su disco, il codice dell'applicazione inizia esattamente al secondo blocco ed è rilocato in maniera da risiedere a partire da $0400.

I puntatori di caricamento e di esecuzione nel File Header sono entrambi impostati con l'indirizzo $0400, assumendo che la routine d'inizializzazione dell'applicazione sia la prima routine dell'applicazione. Se non lo è, il puntatore di esecuzione dev'essere aggiornato di conseguenza. Creato il file sorgente, bisogna compilarlo e salvarlo su disco come un normale file PRG del C-64. Nel notes di deskTop appare con l'icona dei file C-64. Il blocco File Header si trova nel primo settore del file, mentre il codice dell'applicazione inizia dal secondo settore.

Il programma Basic che presentiamo nel prossimo capitolo, trasforma il file in un file GEOS. I byte 19 e 20 del File Entry in directory punteranno al blocco File Header, e i byte 1 e 2 punteranno al primo blocco dell'applicazione. A trasformazione avvenuta, l'icona associata al nuovo file apparirà in deskTop. Premendo il pulsante del mouse due volte sull'icona, si avrà il caricamento in memoria dell'applicazione all'indirizzo specificato nel File Header, e la sua esecuzione a partire dall'indirizzo d'esecuzione contenuto anch'esso nel File Header. La nostra applicazione sarà caricata a $0400 e mandata in esecuzione allo stesso indirizzo.

Nel prossimo capitolo presenteremo prima di tutto la struttura sommaria del file da convertire e il programma Basic di trasformazione in file GEOS. Il File Header è il primo blocco, seguito da due istruzioni fittizie per indicare l'inizio del codice dell'applicazione. Una descrizione completa del File Header si trova nel capitolo dedicato ai file in ambiente GEOS.

3 FILE GEOS IN FORMATO PRG

Introduzione

La semplice applicazione che realizzeremo inizia a $0400, ed è preceduta da un gruppo di dati che costituisce il blocco File Header previsto da GEOS. Tale blocco di dati è lungo 252 byte; i restanti quattro che appariranno all'inizio del primo settore del file sul disco non sono riportati nel file sorgente. Questi quattro byte non possono essere impostati in fase di compilazione in quanto il compilatore li aggiorna in maniera autonoma. I primi due contengono l'indirizzo T/S del successivo blocco del file (oppure $00 seguito dal numero di byte del settore per individuare la fine del file), e questo formato, adottato per connettere sequenzialmente i blocchi di un file, viene impiegato anche in ambiente GEOS. Quando il programma Basic di conversione preleva il primo blocco del file per allocarlo separatamente come blocco File Header, questi due byte sono impostati rispettivamente a $00 e $FF, per indicare che non seguono altri blocchi di dati e tutti i byte del blocco corrente sono significativi. Gli altri due byte del primo blocco del file oggetto salvato su disco, ovvero la seconda coppia, contengono l'indirizzo al quale il file deve risiedere in memoria. Questa informazione non ha significato per il blocco File Header in se stesso, prima di tutto perché questo blocco non deve risiedere in un'area particolare di memoria, e in secondo luogo perché in ambiente GEOS l'indirizzo che individua l'area di memoria che dev'essere occupata da un file viene indicato altrove. Nel blocco File Header, anziché l'indirizzo, questi due byte contengono le dimensioni in altezza (21 pixel) e in larghezza (3 byte) dell'icona del file, che fortunatamente sono sempre le stesse. Quindi il programma Basic non incontra problemi per riscrivere questi due byte con le dimensioni 3 e 21 dell'icona.

Il byte successivo nel blocco File Header contiene il numero di byte che definiscono il disegno dell'icona con il bit 7 impostato a 1 (il bit 7 è un flag di servizio per le routine di GEOS che disegnano le icone). Seguono 63 byte di dati grafici per definire il disegno

dell'icona. Dopo questo gruppo di dati, vi è un byte che contiene il tipo di file secondo lo standard C-64, che per le applicazioni GEOS compatibili dev'essere USR, e un altro byte che indica il tipo secondo lo standard GEOS, che per la nostra applicazione è APPLICATION. Segue il tipo di struttura del file in standard GEOS, che per la nostra applicazione è SEQUENTIAL, i tre puntatori (ognuno di due byte) che indicano rispettivamente l'inizio e la fine del file in memoria, e l'indirizzo dove viene a risiedere la routine di inzializzazione dell'applicazione. Quindi troviamo il nome del file e il nome dell'autore. La struttura del blocco File Header è ampiamente illustrata nel capitolo dedicato al sistema dei file, dove affronteremo anche il significato dei dati in esso contenuti.

TestApplication

Chiamata da: deskTop

Sinossi: Questo file contiene l'applicazione d'esempio. È scritto per essere compilato con un compilatore Assembly standard. I primi 252 byte di questo file contengono i dati che compongono il blocco File Header dell'applicazione e, a compilazione avvenuta, riempiranno interamente il primo blocco del file. Il file può essere convertito in standard GEOS utilizzando un programma Basic separato.

Questo programma Basic separa il primo blocco del file oggetto dell'applicazione e lo alloca separatamente come File Header. Nel File Entry del file presente in directory, due byte contengono l'indirizzo traccia/settore del blocco File Header. I puntatori al primo blocco del file contenuti nel File Entry sono aggiornati con la traccia e il settore nel quale si trova il secondo blocco del file. In questo modo il primo blocco del file diventa il File Header e l'applicazione inizia con il codice oggetto.

```
.psect $0304       ;indirizzo al quale deve inziare la compilazione del codice sorgente

FileHeader:        ;I primi 4 byte come appaiono sul disco dopo che il file e' stato
                   ;trasformato. Questi byte non sono presenti nel file sorgente
                   ;e quindi sono commentati a parte
                   ;.byte       $00, $FF     ;il File Header e' composto da un solo settore
                   ;                         ;e tutti i byte del settore sono significativi
                   ;.byte       3            ;l'icona e' larga 3 byte
                   ;.byte       21           ;l'icona e' alta 21 pixel
```

```
.byte    (63|$80)                              ;64 byte per definire il disegno dell'icona
.byte    %11111111,%11111111,%11111111
.byte    %10000000,%00000000,%00000001
.byte    %10000000,%00000000,%00000001
.byte    %10000000,%00000000,%00000001
.byte    %10011111,%11110000,%00000001
.byte    %10000001,%00000000,%00000001
.byte    %10000001,%00000000,%00000001
.byte    %10000001,%00011110,%00000001
.byte    %10000001,%00100001,%00000001
.byte    %10000001,%00101110,%00000001
.byte    %10000001,%00100000,%00000001
.byte    %10000001,%00011110,%00000001
.byte    %10000001,%00000000,%00000001
.byte    %10000001,%00011111,%00100001
.byte    %10000001,%00100000,%01111101
.byte    %10000001,%00011110,%00100001
.byte    %10000001,%00000001,%00100001
.byte    %10000001,%00111110,%00011101
.byte    %10000000,%00000000,%00000001
.byte    %10000000,%00000000,%00000001
.byte    %11111111,%11111111,%11111111
.byte    $80|USR                               ;tipo Commodore associato ai file GEOS
.byte    APPLICATION                           ;tipo GEOS associato al file
.byte    SEQUENTIAL                            ;tipo struttura del file
.word    LoadAddress                           ;indirizzo a cui allocare il file in memoria
.word    EndCode                               ;indirizzo della fine del file in memoria
.word    InitCode                              ;indirizzo dell'inizio del codice
                                               ;dell'applicazione
.byte    "Test Appl   V1.0",0,0,0,0            ;16 byte filename + 4 zeri
.byte    "Mike Farr",0,0,0,0,0,0,0,0,0,0,0     ;20 byte nome autore o, se file dati, nome
                                               ;disco dell'applicazione parente
.byte    0,0,0,0,0,0,0,0,0,0                   ;20 byte nome applicazione parente
.byte    0,0,0,0,0,0,0,0,0,0                   ;
.byte    0,0,0,0,0,0,0,0,0,0                   ;23 byte liberi per le applicazioni
.byte    0,0,0,0,0,0,0,0,0,0                   ;
.byte    0,0,0                                 ;
.byte    0                                     ;inizio del testo opzionale associato
                                               ;al file, zero per nessun testo
.block   95                                    ;spazio disponibile per il testo opzionale
```

```
LoadAddress:        ;$0400, i dati che iniziano a LoadAddress sono salvati su disco a partire
                    ;dal secondo blocco del file oggetto
InitCode:           ;l'indirizzo InitCode dove risiede la routine d'inizializzazione
                    ;dell'applicazione non deve necessariamente corrispondere
                    ;(come invece avviene in questo caso) a LoadAddress

      lda     #0           ;istruzioni fittizie per indicare l'inizio del codice
      sta     r0L
      ...
```

Il file Basic PRGTOGEOS

In questo paragrafo parleremo del programma Basic che si utilizza per convertire un file PRG in un file GEOS. Per prima cosa il programma chiede in input il nome del file da convertire e la data. GEOS utilizza i byte dal 23 al 27 del File Entry per memorizzare sul programma la data dell'ultima modifica. Altri byte sono trasferiti dal blocco File Header al File Entry del file. Il formato del File Entry, modificato da GEOS, appare nella tavola seguente.

Formato del File Entry di un file

Byte	Descrizione
0	Tipo C-64: 0=DELeted, 1=SEQuential, 2=PRoGram, 3=USeR, 4=RELative. Bit 6=1 file protetto
1-2	Traccia e settore del primo blocco del file
3-18	Nome del file di 16 caratteri. Il nome e' riempito con gli spazi ottenuti con il tasto SHIFT (codice $A0)
19-20	Traccia e settore del blocco File Header
21	Tipo struttura del file formato GEOS: 0=SEQuential, 1=VLIR
22	Tipo file formato GEOS: 0=NOT_GEOS, 1=BASIC, 2=ASSEMBLY, 3=DATA, 4=SYSTEM, 5=DESK_ACC, 6=APPLICATION, 7=APPL_DATA, 8=FONT, 9=PRINTER, 10=INPUT_DEVICE, 11=DISK_DEVICE...
23	Data: anno dell'ultima modifica. Solo le ultime due cifre
24	Data: mese dell'ultima modifica (1-12)
25	Data: giorno dell'ultima modifica (1-31)
26	Data: ora dell'ultima modifica (0-23)
27	Data: minuto dell'ultima modifica (0-59)
28/29	Numero di blocchi (settori) occupati dal file

Il primo blocco del file da convertire memorizzato su disco contiene il File Header dell'applicazione. Questo programma converte il file in formato GEOS.

Il nome del file da convertire viene richiesto in input e non può essere più lungo di 16 caratteri. Ottenuta questa informazione, e la data, il programma cerca il file sul disco.

Quando il file viene trovato, il programma ne separa il File Header (il primo blocco del file) e memorizza nell'apposito puntatore presente nel File Entry la traccia e il settore nel quale si trova. Il puntatore T/S contenuto nel File Entry per indicare il primo blocco del file è aggiornato per puntare alla traccia e al settore del secondo blocco del file (dove ha realmente inizio l'applicazione).

I primi due byte del blocco File Header vengono cambiati in $00 e $FF e gli altri due in 3 e 21 (larghezza in byte e altezza in pixel dell'icona associata al file).

I byte 21 e 22 nel File Entry vengono modificati per indicare rispettivamente il tipo di struttura e il tipo di file in ambiente GEOS. La data dell'ultima modifica è inserita nei byte dal 23 al 27. Il numero di blocchi del file non viene alterato.

```
1000    INPUT "NOME DEL PROGRAMMA";F$          ;input nome dell'applicazione
1010    IF F$="" GOTO 1490                     ;nome applicazione nullo, esci
1020    INPUT "ANNO            (EX: 88)";Y     ;input data e ora
1030    INPUT "MESE            (EX:  5)";MO
1040    INPUT "GIORNO          (EX: 31)";DA
1050    INPUT "ORA             (EX: 14)";H
1060    INPUT "MINUTI          (EX: 35)";MI
1070    OPEN 15, 8, 15, "I:0"                  ;inizializza il drive
1080    OPEN  2, 8,  2, "#"                    ;apre un canale ad accesso diretto
1090    T$=CHR$(18): S$=CHR$(1)                ;traccia 18 settore 1: primo blocco
                                               ;della directory
1100    GOSUB 1520                             ;carica nel buffer un blocco della directory
1110    GET#2, NT$, NS$                        ;preleva la traccia e il settore del blocco
                                               ;successivo
1120    FOR E=0 TO 7                           ;controlla ogni File Entry
1130    GOSUB 1600                             ;preleva il nome del file
1140    IF D$=F$ GOTO 1180                     ;se il nome del file corrisponde
1150    NEXT E
1160    IF NT$="" GOTO 1490                    ;directory terminata, nome non trovato,
                                               ;esci
1170    T$=NT$: S$=NS$: GOTO 1100              ;carica il blocco successivo
1180    DT$=T$: DS$=S$                         ;il nome e' stato trovato
1190    T$=HT$: S$=HS$: GOSUB 1520             ;preleva il blocco File Header
1200    GET#2, MT$, MS$                        ;preleva T/S per il secondo blocco
1210    FOR I=2 TO 67                          ;salta al tipo di file del C-64
```

```
1220    GET#2, B$
1230    NEXT I
1240    GET#2, CT$, GT$, GF$                 ;preleva CT$ = tipo file C-64, GT$ = tipo
                                             ;file GEOS, GF$ = struttura file GEOS
1250    GOSUB 1520                           ;preleva nuovamente il blocco File Header
1260    PRINT#2, CHR$(0); CHR$(255);         ;scrive $00 e $FF nei byte T/S
1270    PRINT#2, CHR$(3); CHR$(21) ;         ;scrive 3,21
1280    GOSUB 1560                           ;riscrive il blocco File Header
1290    T$=DT$: S$=DS$: GOSUB 1520           ;preleva il blocco della directory
                                             ;che contiene il File Entry del file
1300    FOR I=0 TO 32*E+1                    ;salta al File Entry numerato da E
1310    GET#2, B$
1320    NEXT I
1330    IF MS$="" THEN MS=0: GOTO 1350       ;se il settore e'  0  l'istruzione asc() da'
                                             ;errore
1340    MS=ASC(MS$)
1350    PRINT#2, CT$; MT$; CHR$(MS);         ;memorizza il tipo C-64 e il puntatore T/S
                                             ;del primo blocco utile del file (il secondo
                                             ;blocco) nel File Entry
1360    FOR I=1 TO 16                        ;salta alla T/S del File Header
1370    GET#2, B$
1380    NEXT I
1390    IF HS$="" THEN HS=0: GOTO 1410
1400    HS=ASC(HS$)
1410    IF GF$="" THEN GF=0: GOTO 1430
1420    GF=ASC(GF$)
1430    IF GT$="" THEN GT=0: GOTO 1450
1440    GT=ASC(GT$)
1450    PRINT#2,HT$;CHR$(HS);CHR$(GF);CHR$(GT);          ;memorizza T/S del File Header,
                                                         ;struttura GEOS del file (VLIR
                                                         ;o SEQ), il tipo file GEOS
1460    PRINT#2,CHR$(Y);CHR$(MO);CHR$(DA);   ;memorizza la data
1470    PRINT#2,CHR$(H);CHR$(MI);            ;memorizza l'ora
1480    GOSUB 1560                           ;riscrive il File Header su disco
1490    CLOSE 2
1500    CLOSE 15
1510    END
1520    IF S$="" THEN SE=0: GOTO 1540
1530    SE=ASC(S$)
1540    PRINT#15, "U1"; 2; 0; ASC(T$); SE    ;subroutine per prelevare un blocco dal disco
1550    RETURN
1560    IF S$="" THEN SE=0: GOTO 1580
```

```
1570    SE=ASC(S$)
1580    PRINT#15, "U2"; 2; 0; ASC(T$); SE    ;subroutine per scrivere un blocco sul disco
1590    RETURN
1600    REM                                  ;subroutine per cercare il nome del file
1610    D$=""                                ;inizializza il nome del file
1620    GET#2, B$: I=1                       ;preleva il primo carattere, inizializza I
1630    IF B$="" GOTO 1690                   ;File Entry nullo
1640    IF ASC(B$) <> 130 GOTO 1690          ;se il tipo non e' PRG allora vai
                                             ;alla riga 1690
1650    GET#2, HT$, HS$: I=3                 ;T/S del primo blocco del file
                                             ;(il File Header)
1660    GET#2, B$: I=I+1                     ;preleva un carattere dal nome del file
1670    IF ASC(B$)=160 GOTO 1690             ;se e' alla fine del nome salta  al prossimo
                                             ;File Entry
1680    D$=D$+B$: GOTO 1660                  ;preleva il successivo carattere del nome
1690    FOR I=I TO 31                        ;salta il File Entry corrente
1700    GET#2, B$
1710    NEXT I
1720    RETURN
```

Pronti alla creazione di un file GEOS

A questo punto il programmatore possiede gli strumenti di lavoro fondamentali per realizzare una semplice applicazione come quella che vi proponiamo. È importante effettuare qualche prova di conversione di file PRG, in maniera da familiarizzarsi con questa operazione. Il nostro consiglio è di iniziare a scrivere, compilare e trasformare un'applicazione molto semplice, e approfondire un po' alla volta le caratteristiche offerte da GEOS inserendole in nuove versioni più complesse della nostra applicazione.

Ognuno dei prossimi capitoli tratterà un argomento specifico. Dopo averli letti e approfonditi, il programmatore dovrebbe fare qualche piccola esperienza aggiungendo funzioni al piccolo programma che stiamo per descrivere. In questo modo, procedendo nella lettura del libro, si acquisisce gradualmente familiarità con il sistema nei suoi molteplici aspetti.

Il file QuantumTest

Questa applicazione si propone di illustrare la gestione dei menu e delle icone.

```
.include geosConstants                       ;simboli delle costanti
.include geosMemoryMap                       ;simboli delle variabili
```

```
.include geosRoutines                       ;simboli delle routine
.include geosMacros                         ;file contenente le macro istruzioni
                                            ;utilizzate nella compilazione

USE_PRGTOGEOS = 1                  ;flag di compilazione per indicare se viene usato
                                   ;(valore 1) o non viene usato (valore 0) il programma
                                   ;PRGTOGEOS di conversione

.if USE_PRGTOGEOS                  ;se il valore del flag e' 1 il primo blocco del
                                   ;file e' il File Header associato

.psect $0304                       ;il testo viene compilato a partire da questo indirizzo
```

Il blocco File Header dell'applicazione

Sinossi: Il blocco File Header contiene varie informazioni associate al file. Fra queste assumono particolare importanza l'indirizzo di allocazione in memoria, l'indirizzo di esecuzione, il nome del file e il disegno dell'icona associata.

Se si adotta il metodo di trasformazione del file PRG in SEQUENTIAL standard GEOS, impiegando il programma Basic PRGTOGEOS, i dati che compongono il File Header devono, a compilazione avvenuta, occupare interamente il primo blocco del file oggetto. I primi quattro byte sono commentati a parte come si spiega nell'introduzione al capitolo.

```
FileHeader:     ;inizio dei dati che costituiscono il blocco File Header
                ;.byte        $00,$FF  ;il File Header e' composto da un solo settore
                ;.byte        3        ;l'icona e' larga 3 byte
                ;.byte        21       ;l'icona e' alta 21 pixel

.byte    (63|$80) ;64 byte per definire il disegno dell'icona
.byte    %11111111,%11111111,%11111111
.byte    %10000000,%00000000,%00000001
.byte    %10000000,%00000000,%00000001
.byte    %10000000,%00000000,%00000001
.byte    %10011111,%11110000,%00000001
```

```
.byte   %10000001,%00000000,%00000001
.byte   %10000001,%00000000,%00000001
.byte   %10000001,%00011110,%00000001
.byte   %10000001,%00100001,%00000001
.byte   %10000001,%00101110,%00000001
.byte   %10000001,%00100000,%00000001
.byte   %10000001,%00011110,%00000001
.byte   %10000001,%00000000,%00000001
.byte   %10000001,%00011111,%00100001
.byte   %10000001,%00100000,%01111101
.byte   %10000001,%00011110,%00100001
.byte   %10000001,%00000001,%00100001
.byte   %10000001,%00111110,%00011101
.byte   %10000000,%00000000,%00000001
.byte   %10000000,%00000000,%00000001
.byte   %11111111,%11111111,%11111111
.byte   $80|USR                          ;tipo Commodore associato ai file GEOS
.byte   APPLICATION                      ;tipo GEOS associato al file
.byte   SEQUENTIAL                       ;tipo struttura del file
.word   QuantumTest                      ;indirizzo ove locare il file in memoria
.word   EndCode                          ;indirizzo della fine del file in memoria
.word   InitCode                         ;indirizzo dell'inizio del codice
                                         ;dell'applicazione
.byte   "Quantum     V1.0",0,0,0,0       ;16 byte filename + 4 zeri
.byte   "Mike Farr",0,0,0,0,0,0,0,0,0,0,0 ;20 byte nome autore
.byte   0,0,0,0,0,0,0,0,0,0              ;20 byte nome applicazione parente
.byte   0,0,0,0,0,0,0,0,0,0              ;
.byte   0,0,0,0,0,0,0,0,0,0              ;23 byte liberi per l'applicazione
.byte   0,0,0,0,0,0,0,0,0,0              ;
.byte   0,0,0                            ;
.byte   0                                ;inizio del testo opzionale associato
                                         ;al file, zero per nessun testo
.block  95                               ;spazio disponibile per il testo opzionale


.else                                    ;fine del blocco File Header


.psect  $0400                            ;nel caso non venga impiegato il programma
                                         ;PRGTOGEOS, il codice oggetto deve iniziare
                                         ;a $0400
.endif
```

File GEOS di dimostrazione Quantum Link. Routine principale

Autore: Mike Farr

Chiamata da: deskTop

Parametri: Nessuno

Restituisce: Niente

Distrugge: Orientativamente, tutti i registri

Sinossi: Questo modulo contiene alcune routine atte a verificare il funzionamento di menu e icone. Quando un file è mandato in esecuzione da deskTop, il controllo del sistema viene ceduto all'indirizzo indicato dalla label InitCode. L'applicazione definisce e visualizza il primo schermo utilizzando le appropriate routine.

Il primo schermo mostra otto icone (corrispondenti a otto voci del menu principale della rete Q-link). I disegni delle icone possono essere di qualunque dimensione. Le piccole icone che utilizziamo in questa applicazione sono tratte dall'applicazione geoPaint e in effetti sono un po' piccole per uno schermo iniziale.

Tutte le icone visualizzate sul primo schermo eseguono la stessa routine di servizio. Quando una viene attivata, la routine di servizio cancella lo schermo corrente e visualizza la struttura del menu principale del secondo schermo.

Il secondo schermo mostra la struttura di un menu come potrebbe apparire nella rete Q-link. I nomi che compaiono nelle voci dei menu possono essere sostituiti da qualsiasi parola o frase significativa per l'utente, e tutte le voci dei sotto-menu chiamano la stessa routine di servizio. Questa routine comune a tutte le voci, non fa altro che chiamare la routine di sistema GotoFirstMenu per lasciare attivato solo il menu principale e disattivare tutta la struttura di sotto-menu correntemente aperta.

L'icona nella parte bassa a sinistra del secondo schermo restituisce il controllo a deskTop, e quindi determina la conclusione dell'applicazione.

```
QuantumTest:                              ;inizio codice dell'applicazione
InitCode:                                 ;routine principale attivata per eseguire
                                          ;l'applicazione

        jsr     NewDisk                   ;un bug presente in precedenti versioni
                                          ;di GEOS rende necessario chiamare NewDisk
                                          ;per fermare il motorino di rotazione del
                                          ;disco se l'applicazione non accede al drive
        jsr     MouseUp                   ;attiva il mouse (inutile se e' gia' attivato)
        lda     #2
        jsr     SetPattern                ;imposta la matrice grafica punteggiata al 50%

        jsr     i_Rectangle               ;cancella lo schermo riempiendolo con
                                          ;la matrice grafica prescelta
.byte   0
.byte   199
.byte   0
.byte   319

        LoadW   r0, QuantumTestIcons      ;indirizzo della tavola di dati che
                                          ;definisce le icone da visualizzare
        jsr     DoIcons                   ;definisce le icone
        rts
```

Tavola di definizione delle icone

Chiamata da: La routine principale QuantumTest cede il controllo della tavola di definizione a MainLoop tramite la chiamata alla routine Dolcons.

Parametri: Nessuno

Restituisce: Niente

Sinossi: La tavola definisce le otto icone del primo schermo.

```
Y_POS_TOP_ICON = 10             ;le coordinate x e y sono misurate dall'angolo sinistro
                                ;in alto dello schermo,e individuano sempre l'angolo
                                ;superiore sinistro dell'icona
X_POS_TOP_ICON = 3

QuantumTestIcons:
.byte    8                      ;numero di icone
.word    160                    ;coordinata x della posizione del mouse dopo
                                ;la visualizzazione delle icone
.byte    100                    ;coordinata y della posizione del mouse dopo
                                ;la visualizzazione delle icone

.word    showCaseData           ;puntatore ai dati grafici per l'icona showCase
.byte    X_POS_TOP_ICON         ;coordinata x in byte dell'icona
.byte    Y_POS_TOP_ICON         ;coordinata y in pixel dell'icona
.byte    2, 16                  ;larghezza in byte e altezza in pixel dell'icona
.word    DoShow                 ;routine di servizio per l'icona showCase

.word    justForFunData         ;puntatore ai dati grafici per`l'icona hand
.byte    X_POS_TOP_ICON         ;coordinata x in byte dell'icona
.byte    Y_POS_TOP_ICON+30      ;coordinata y in pixel dell'icona
.byte    2, 16                  ;larghezza in byte e altezza in pixel dell'icona
.word    DoFun                  ;routine di servizio per l'icona hand

.word    custServData           ;puntatore ai dati grafici per l'icona eraser
.byte    X_POS_TOP_ICON         ;coordinata x in byte dell'icona
.byte    Y_POS_TOP_ICON+60      ;coordinata y in pixel dell'icona
.byte    2, 16                  ;larghezza in byte e altezza in pixel dell'icona
.word    DoServ                 ;routine di servizio per l'icona eraser

                                ;omettiamo i commenti per i gruppi di definizione delle ul-
                                ;time cinque icone, in quanto la struttura rimane identica

.word    mallData
.byte    X_POS_TOP_ICON+15
.byte    Y_POS_TOP_ICON+60
.byte    2, 16
.word    DoMall
```

```
.word   newsData
.byte   X_POS_TOP_ICON+30
.byte   Y_POS_TOP_ICON+60
.byte   2, 16
.word   DoNews

.word   learningData
.byte   X_POS_TOP_ICON+30
.byte   Y_POS_TOP_ICON+30
.byte   2, 16
.word   DoLearn

.word   cinData
.byte   X_POS_TOP_ICON+30
.byte   Y_POS_TOP_ICON
.byte   2, 16
.word   DoCin

.word   pConData
.byte   X_POS_TOP_ICON+15
.byte   Y_POS_TOP_ICON
.byte   2, 16
.word   DoPCon
```

Routine di servizio per le icone del primo schermo

Chiamata da: Attivazione delle icone del primo schermo.

Parametri: Nessuno

Restituisce: Niente

Sinossi: Attivando una delle icone del primo schermo, viene eseguita la routine di servizio associata. Nel nostro esempio tutte le icone del primo schermo eseguono la stessa routine di servizio, che cancella lo schermo corrente e attiva il successivo.

”

```
DoShow:                         ;routine di servizio per l'icona
                                ;Commodore Software Showcase
```

```
DoFun:                    ;routine di servizio per l'icona Just For Fun
DoServ:                   ;routine di servizio per l'icona Customer Service Center
DoMall:                   ;routine di servizio per l'icona The Mall
DoNews:                   ;routine di servizio per l'icona News and Information
DoLearn:                  ;routine di servizio per l'icona Learning Center
DoCin:                    ;routine di servizio per l'icona Commodore Information
                          ;Network
DoPCon:                   ;routine di servizio per l'icona People Connection

 jsr    i_Rectangle       ;cancella lo schermo
 .byte  0
 .byte  199
 .word  0
 .word  319

 LoadW  r0, Screen2Icon   ;indirizzo della tavola di definizione dell'icona
                          ;presente sullo schermo
 jsr    DoIcons           ;attiva le icone. Per via di un bug presente in GEOS,
                          ;dev'essere definita (ma non necessariamente
                          ;visualizzata) almeno un'icona
 lda    #0                ;posiziona il mouse sulla prima voce del menu
 LoadW  r0, QuantumTestMenu ;indirizzo della tavola di definizione dei menu
 jsr    DoMenu            ;definisce i menu

 rts
```

Dati grafici delle icone del primo schermo

Sinossi: Queste icone sono state tratte dall'applicazione geoPaint.

```
showCaseData:
.byte   $02,$FF,$9C,$80,$01,$80,$39,$80,$6D,$80,$E5,$81,$BD,$83,$19,$86
.byte   $31,$8C,$61,$98,$C1,$B1,$81,$BB,$01,$BE,$01,$BC,$01,$80,$01,$02
.byte   $FF
```

```
justForFunData:
.byte   $02,$FF,$9C,$87,$01,$8D,$81,$9E,$C1,$BB,$61,$AD,$B1,$B6,$99,$9B
.byte   $09,$9D,$0D,$96,$07,$9A,$03,$8C,$01,$87,$E1,$80,$31,$80,$19,$02
.byte   $FF
custServData:
.byte   $02,$FF,$9C,$80,$01,$80,$01,$BF,$01,$BF,$81,$BF,$C1,$AF,$E1,$A7
.byte   $F1,$B3,$F1,$9A,$11,$8E,$11,$86,$11,$83,$F1,$80,$01,$80,$01,$02
.byte   $FF
mallData:
.byte   $02,$FF,$9C,$80,$01,$80,$01,$83,$81,$83,$81,$86,$C1,$86,$C1,$84
.byte   $C1,$8C,$61,$88,$61,$8F,$E1,$98,$31,$90,$31,$B8,$79,$80.$01,$02
.byte   $FF
newsData:
.byte   $02,$FF,$8E,$80,$01,$80,$01,$90,$01,$98,$01,$8C,$01,$86,$01,$83
.byte   $01,$02,$81,$8C,$80,$C1,$80,$61,$80,$31,$80,$19,$80,$09,$80,$01
.byte   $FF
learningData:
.byte   $02,$FF,$9C,$80,$01,$80,$01,$9D,$B9,$90,$09,$90,$09,$80,$01,$90
.byte   $09,$90,$09,$80,$01,$90,$09,$90,$09,$9D,$B9,$80,$01,$80,$01,$02
.byte   $FF
cinData:
.byte   $02,$FF,$9C,$80,$01,$80,$01,$9D,$B9,$90,$09,$90,$09,$80,$01,$90
.byte   $09,$90,$09,$80,$01,$90,$09,$90,$09,$9D,$B9,$80,$01,$80,$01,$02
.byte   $FF
pConData:
.byte   $02,$FF,$9C,$80,$01,$80,$01,$9D,$B9,$90,$09,$90,$09,$80,$01,$90
.byte   $09,$90,$09,$80,$01,$90,$09,$90,$09,$9D,$B9,$80,$01,$80,$01,$02
.byte   $FF
```

Tavola di definizione dell'icona del secondo schermo

Chiamata da: Viene utilizzata dalla routine DoIcons che a sua volta viene eseguita dalla routine di servizio associata alle icone del primo schermo.

Sinossi: Definisce l'icona nell'angolo inferiore sinistro dello schermo la cui selezione termina l'esecuzione dell'applicazione e ricarica deskTop.

99

```
Screen2Icon:
        .byte   1               ;numero di icone
        .word   240             ;coordinata x del mouse una volta definita l'icona
        .byte   155             ;coordinata y del mouse una volta definita l'icona

        .word   showCaseData    ;utilizza nuovamente i dati grafici dell'icona showCase
        .byte   30              ;coordinata x in byte dell'icona
        .byte   150             ;coordinata y in pixel dell'icona
        .byte   2, 16           ;larghezza in byte e altezza in pixel dell'icona
        .word   ReturnDeskTop   ;routine di servizio associata per terminare l'applicazione

ReturnDeskTop:
        jmp     EnterDeskTop    ;termina l'applicazione ritornando a deskTop
```

Struttura del menu del secondo schermo

```
MAIN_TOP    = 10
MAIN_BOT    = 24                ;altezza del menu orizzontale principale
                                ;corrispondente all'altezza di una linea di testo
MAIN_LFT    = 0
MAIN_RT     = 255               ;valore da determinare sperimentalmente (impostato a 255
                                ;per un menu di dimensione orizzontale massima)

QuantumTestMenu:
  .byte  MAIN_TOP               ;posizione del lato superiore del menu
  .byte  MAIN_BOT               ;posizione del lato inferiore del menu
  .word  MAIN_LFT               ;posizione del lato sinistro del menu
  .word  MAIN_RT                ;posizione del lato destro del menu
  .byte  HORIZONTAL|2           ;tipo di menu OR'ed con il numero di voci di cui
                                ;e' composto

  .word  changeDeptText         ;indirizzo della stringa di testo per questa voce
  .byte  SUB_MENU               ;flag per indicare che la voce attiva un sotto-menu
  .word  ChangeDeptMenu         ;indirizzo della struttura del sotto-menu
  .word  showCaseText           ;indirizzo della stringa di testo per questa voce
```

```
 .byte   SUB_MENU                  ;flag per indicare che la voce attiva un sotto-menu
 .word   ShowCaseMenu              ;indirizzo della struttura del sotto-menu
```

Testi per le voci del menu principale

```
changeDeptText:   .byte    "Change Departments",0               ;testo per questa voce
showCaseText:     .byte    "Commodore Software Showcase",0      ;testo per questa voce
```

Tavola di definizione del sotto-menu "Change Departments"

Chiamata da: Sotto-menu attivato dalla voce "Change Departments" del menu principale del secondo schermo.

Voci del menu:

Commodore Software Showcase	The Mall
Just For Fun	Commodore Information Network
Customer Service Center	Learning Center
People Connection	News and Information

```
ChangeDeptMenu:
 .byte   MAIN_BOT                  ;il lato superiore del sotto-menu inizia in corrispon-
                                   ;denza del lato inferiore del menu principale
 .byte   MAIN_BOT+(8*14)+1         ;il lato inferiore del sotto-menu, e' ottenuto calcolando
                                   ;14 pixel verticali per ogni voce, piu' uno finale
 .word   MAIN_LFT                  ;lato sinistro del sotto-menu
 .word   MAIN_LFT+155              ;lato destro del sotto-menu
 .byte   VERTICAL|8                ;tipo menu OR'ed con il numero di voci

 .word   ShowcaseText              ;puntatore (per questa voce) alla stringa gia'
                                   ;precedentemente inserita nel file sorgente
```

```
ChangeDeptMenu:
 .byte  MAIN_BOT              ;il lato superiore del sotto-menu inizia in corrispondenza
                              ;del lato inferiore del menu principale
 .byte  MAIN_BOT+(8*14)+1     ;il lato inferiore del sotto-menu, e' ottenuto calcolando
                              ;14 pixel verticali per ogni voce, piu' uno finale
 .word  MAIN_LFT              ;lato sinistro del sotto-menu
 .word  MAIN_LFT+155          ;lato destro del sotto-menu
 .byte  VERTICAL|8            ;tipo menu OR'ed con il numero di voci

 .word  showcaseText          ;puntatore (per questa voce) alla stringa
                              ;gia' definita
                              ;precedentemente definita
 .byte  MENU_ACTION           ;flag per indicare una routine di servizio
 .word  ShowcaseDsp           ;indirizzo della routine di servizio associata

 .word  funText               ;puntatore alla stringa per questa voce
 .byte  MENU_ACTION           ;flag per indicare una routine di servizio
 .word  funDsp                ;indirizzo della routine di servizio associata

                              ;per i gruppi di definizione delle ultime sei voci
                              ;del sotto-menu, omettiamo i commenti

 .word  serviceText
 .byte  MENU_ACTION
 .word  ServiceDsp

 .word  peopleConText
 .byte  MENU_ACTION
 .word  PeopleConDsp

 .word  mallText
 .byte  MENU_ACTION
 .word  MallDsp

 .word  infoNetText
 .byte  MENU_ACTION
 .word  InfoNetDsp

 .word  learningText
 .byte  MENU_ACTION
 .word  LearningDsp
```

```
  .word  newsText
  .byte  MENU_ACTION
  .word  NewsDsp
```

Tavola di definizione del sotto-menu "Software Showcase"

Chiamata da: Sotto-menu attivato dalla voce "Commodore Software Showcase" del menu principale del secondo schermo.

Voci del Menu: Software Catalog, SIG Software Library, Software Reviews, Change To Another Department, Software Previews, Person-to-Person File Transfer, Q-Link Post Office

```
ShowcaseMenu:
  .byte  MAIN_BOT                  ;lato superiore del sotto-menu
  .byte  MAIN_BOT+(7*14)+1         ;lato inferiore del sotto-menu calcolato per sette voci
  .word  94                        ;lato sinistro del sotto-menu verticale
  .word  94+140                    ;lato destro del sotto-menu verticale
  .byte  CONSTRAINED|VERTICAL|7    ;tipo di sotto-menu e numero voci. CONSTRAINED
                                   ;significa che il mouse puo' uscire solo dal lato
                                   ;superiore del sotto-menu

  .word  catalogText               ;indirizzo del testo per questa voce
  .byte  MENU_ACTION               ;la prossima word indica l'indirizzo della routine
                                   ;di servizio
  .word  CatalogDsp                ;routine di servizio per questa voce

                                   ;per i prossimi sei gruppi di definizione delle voci,
                                   ;omettiamo i commenti in quanto la loro struttura
                                   ;non cambia

  .word  previewText
  .byte  MENU_ACTION
  .word  PreviewDsp

  .word  sIGLibText
  .byte  MENU_ACTION
  .word  SIGLibDsp
```

```
.word  fileTransText
.byte  MENU_ACTION
.word  FileTransDsp

.word  reviewText
.byte  MENU_ACTION
.word  ReviewDsp

.word  postOfficeText
.byte  MENU_ACTION
.word  PostOfficeDsp

.word  changeText
.byte  MENU_ACTION
.word  ChangeDsp
```

Testo per le voci del sotto-menu "Change Departments"

```
learningText:     .byte        "Learning Center",0
funText:          .byte        "JustForFun",0
serviceText:      .byte        "Customer Service Center",0
infoNetText:      .byte        "Commodore Information Network",0
peopleConText:    .byte        "People Connection",0
newsText:         .byte        "News and Information",0
mallText:         .byte        "The Mall",0
```

Testo per le voci del sotto-menu "Commodore Showcase"

```
catalogText:      .byte        "Software Catalog",0
previewText:      .byte        "Software Previews",0
sIGLibText:       .byte        "SIG Software Library",0
fileTransText:    .byte        "Person-to-Person File Transfer",0
reviewText:       .byte        "Software Reviews",0
postOfficeText:   .byte        "Q-Link Post Office",0
changeText:       .byte        "Change to Another Dept.",0
```

Routine di servizio per le voci dei sotto-menu “Software Showcase” e “Change Departments”

Chiamata da: Dalle voci dei sotto-menu “Software Showcase” e “Change Departments”.

Sinossi: Routine di servizio fittizia per tutte le voci dei sotto-menu.

”

```
LearningDsp:      ;Routine di servizio per il sotto-menu "Change Departments"
FunDsp:
ServiceDsp:
InfoNetDsp:
PeopleConDsp:
NewsDsp:
MallDsp:

CatalogDsp:       ;Routine di servizio per il sotto-menu "Software Showcase"
PreviewDsp:
SIGLibDsp:
FileTransDsp:
ReviewDsp:
PostOfficeDsp:
ChangeDsp:
ShowcaseDsp:

        jsr     GotoFirstMenu ;chiude i sotto-menu
        rts

EndCode:                       ;fine dell'applicazione
```

4 GEOS E LA GRAFICA

Introduzione

Gli argomenti che abbiamo trattato finora sono sufficienti per creare piccole applicazioni e salvarle su disco in file GEOS. È quindi arrivato il momento di parlare delle nuove risorse offerte da GEOS al programmatore, e dare così maggior consistenza alle applicazioni. In questo capitolo scopriamo quali sono le opportunità grafiche disponibili in ambiente GEOS. Come è già stato accennato, GEOS utilizza uno schermo grafico realizzato con il modo bit-map in alta risoluzione installato nel C-64. Non ci soffermeremo a illustrare questo modo grafico, probabilmente già noto a tutti.

Disegnare sullo schermo di GEOS significa impostare a 1, o a 0, i bit della RAM impiegata per definire lo schermo in alta risoluzione. I bit impostati a 1 sono visualizzati con il colore di "foreground", ovvero il colore del tratto, mentre quelli impostati a 0 sono visualizzati con il colore di "background", cioè il colore di fondo. I colori di default in ambiente GEOS sono grigio scuro per il tratto e grigio chiaro per lo sfondo. Questa scelta può apparire insolita dal momento che si è portati a pensare che lo schermo privo di disegni debba essere nero. In realtà questa combinazione di colori nasce dal desiderio di offrire un'interfaccia utente che abbia le sembianze di un foglio di carta bianco sul quale si scrive con una penna nera. In ambiente GEOS lo sfondo vuole essere un foglio bianco e i tratti sono realizzati con una penna nera. Questi colori possono essere cambiati ricorrendo al desk accessory Preference Manager, e quindi risulta più comodo e chiaro parlare di colore di primo piano, o del tratto, e colore di fondo.

Le dimensioni geometriche

Tutte le dimensioni geometriche passate alle routine di gestione della grafica sono inclusive. Con questo termine intendiamo dire che, per esempio, una linea disegnata grazie a informazioni sulle coordinate di due punti sullo schermo, comprende anche il punto d'inizio e il punto d'arrivo, e un rettangolo, come vedremo più avanti, include le linee del bordo.

I colori

Se si desidera gestire in maniera articolata anche i colori, bisogna tener conto delle restrizioni imposte dal sistema di gestione dei colori in bit-map del C-64. Nel momento in cui scriviamo, tutte le routine grafiche disponibili in GEOS non hanno effetto sulla pagina del colore associata allo schermo in bit-map. Quindi qualsiasi figura disegnata con le routine grafiche utilizza solo il colore del tratto e il colore di fondo. È compito dell'applicazione manipolare i colori e predisporre, se necessario, 1000 byte per mantenere una copia della pagina del colore perché ci sia la possibilità di realizzare l'opzione Undo. Questa gestione dei colori autonoma, cioè indipendente dalle routine grafiche di GEOS, è per esempio realizzata nell'applicazione geoPaint.

Il buffer di schermo

Come già brevemente illustrato nel primo capitolo, GEOS gestisce un secondo schermo allocando 8000 byte in un buffer di schermo. Questo buffer è utilizzato per copiare i contenuti dello schermo principale e per conservarli in caso di necessità. Per esempio, all'apertura di un menu la parte di schermo coperta viene momentaneamente salvata nel buffer di schermo, e alla chiusura del menu viene riallocata sullo schermo originale per ripristinare i disegni precedentemente coperti. Un altro impiego del buffer di schermo può essere la realizzazione da parte delle applicazioni della funzione Undo. Se non si desidera salvare momentaneamente parti dello schermo principale, si può disabilitare l'uso del buffer di schermo e utilizzare per altri scopi l'area di memoria rimasta disponibile. Ripristinare una parte dello schermo precedentemente salvata significa copiare una particolare area dal buffer allo schermo principale.

GEOS prevede un flag per indicare se tutte le operazioni grafiche eseguite dall'applicazione (disegni e testi) devono essere effettuate sia sullo schermo principale sia nel buffer di schermo, oppure solo sullo schermo principale o solo nel buffer di schermo. Il nome di questo flag è dispBufferOn. GEOS accede ai bit 6 e 7 di tale byte per determinare la configurazione dello schermo:

```
dispBufferOn        bit 7 - se impostato a 1 scrive sullo schermo principale
                    bit 6 - se impostato a 1 scrive nel buffer di schermo
```

Le routine grafiche, come quelle per il disegno di linee che analizzeremo fra poco, per eseguire i loro compiti fanno riferimento al valore contenuto nel flag dispBufferOn. Il modo più comune per impiegare dispBufferOn, consiste nel limitare il disegno di una forma geometrica allo schermo principale e utilizzare il contenuto precedente del buffer di schermo per cancellarla. Lo stesso GEOS ricorre a questa tecnica per gestire i menu e i box di dialogo. Il menu viene disegnato solo sullo schermo principale, e quando l'utente lo chiude GEOS lo cancella copiando una parte del buffer di schermo sullo schermo principale. GEOS supporta routine specifiche per trasferire parti dello schermo principale nel buffer di schermo e viceversa.

Alcune applicazioni che necessitano di una maggiore quantità di memoria, possono destinare gli 8K del buffer di schermo a contenere i loro codici e i dati. Ma in questo modo sorge il problema di come gestire i menu, i box di dialogo e i desk accessory. Dev'essere l'applicazione, allora, a provvedere autonomamente al ripristino dello schermo ogni volta che occorre.

GEOS prevede un vettore, recoverVector, che normalmente contiene l'indirizzo della routine grafica RecoverRectangle (la vedremo più avanti). Ogni volta che GEOS deve ripristinare lo schermo coperto da un menu, un box di dialogo o un desk accessory, prepara dei parametri per indicare le dimensioni del rettangolo da copiare dal buffer sullo schermo principale. Questi parametri serviranno alla routine RecoverRectangle, che GEOS chiama attraverso il vettore recoverVector. Se l'applicazione utilizza per altri scopi la memoria destinata di solito al buffer di schermo, allora deve anche provvedere all'aggiornamento di recoverVector con l'indirizzo della routine appositamente creata per ripristinare lo schermo.

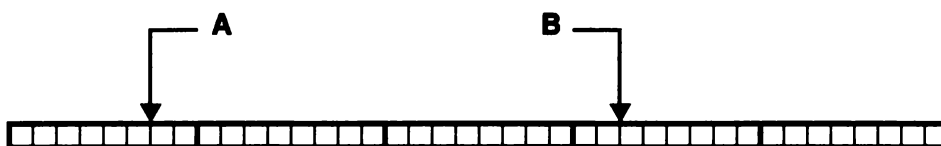
Se l'applicazione memorizza nel vettore recoverVector l'indirizzo di una routine propria, questa potrà accedere a tutti i parametri che normalmente GEOS prepara per la routine RecoverRectangle. Quindi se un'applicazione utilizza il buffer di schermo per impieghi particolari, deve prevedere una routine in grado, all'occorrenza, di ripristinare lo schermo.

Il disegno delle linee

Iniziamo ad approfondire le capacità grafiche di GEOS illustrando la forma più semplice di disegno: la linea. Prima di proseguire è bene distinguere le linee verticali e orizzontali da quelle diagonali, dal momento che il modo di trattarle è sostanzialmente diverso. Le linee orizzontali e verticali possono essere disegnate secondo una matrice grafica di riferimento, cioè non devono essere necessariamente continue, mentre le linee diagonali lo sono sempre.

Quando GEOS disegna una linea orizzontale, il byte che rappresenta la matrice grafica, o di continuità, della linea viene ripetutamente memorizzato nei byte della RAM che contengono lo schermo grafico. Ogni matrice grafica copre esattamente un byte della memoria. Questo significa che non viene traslata di alcun bit, ed è quindi

indipendente dagli estremi della linea. Per rendere indipendente la collocazione del byte della matrice grafica dalla posizione della linea orizzontale sullo schermo, GEOS provvede a mascherare alcuni bit del byte in maniera che la visualizzazione della matrice di continuità della linea inizi proprio dove inizia la linea e termini esattamente ove la linea termina. Dal momento che la matrice di continuità viene poi ripetuta orizzontalmente per ottenere la linea, nell'ultima ripetizione sarà troncata per visualizzare l'esatta lunghezza della linea. L'effetto prodotto dal costante allineamento del byte della matrice grafica con gli spazi carattere dello schermo, cioè la sua indipendenza dai limiti della linea, si osserva per esempio disegnando due linee orizzontali una di fianco all'altra con diversa lunghezza e stessa matrice grafica: hanno evidentemente lo stesso disegno di continuità e i due disegni sono allineati. I bit a 0 e i bit a 1 delle due linee, essendo queste prodotte con la stessa matrice grafica, sono allineati verticalmente.

Per esempio, supponiamo che GEOS disegni una linea dal punto A al punto B della linea di scansione:

con la seguente matrice di continuità:

Il byte della matrice di continuità è sempre allineato in modo tale che un suo bit, per esempio il bit 0, appaia sempre nel bit 0 di ogni byte della RAM allocata per lo schermo grafico. Per iniziare e terminare la linea nei punti prefissati, vengono mascherati gli opportuni bit del byte della matrice grafica d'inizio e di fine della linea:

Le linee verticali sono disegnate in modo simile. In questo caso il byte della matrice di continuità è disposto verticalmente e scritto sullo schermo dall'alto in basso. Anche per le linee verticali vale lo stesso discorso di allineamento delle matrici con gli spazi carattere in maniera indipendente dagli estremi della linea. Il tempo impiegato per disegnare una linea verticale è maggiore di quello necessario per le linee orizzontali, in quanto GEOS deve fare un maggior numero di operazioni per disegnare il byte della matrice grafica su otto diverse linee di scansione. Questo significa che il bit 0 della matrice grafica si trova in corrispondenza di un certo bit del primo byte, che dipende dalla coordinata x della linea, e il bit 1 si trova sul byte appena sottostante, sempre all'altezza dello stesso bit. È quindi evidente che i bit della matrice grafica devono essere ripartiti in 8 byte diversi, e da questo deriva la relativa lentezza del processo. Scrivere un bit all'interno di un byte significa effettuare l'operazione logica OR fra il bit e il byte. Il tempo impiegato per disegnare una linea verticale è circa otto volte maggiore del tempo necessario per una linea orizzontale.

Le linee diagonali

Il procedimento di visualizzazione delle linee diagonali è diverso da quello utilizzato per le linee orizzontali e verticali. Le linee diagonali non vengono visualizzate utilizzando una matrice di continuità. GEOS le disegna "a piena continuità", cioè ogni bit della linea è impostato a 1. Non è possibile indicare una matrice di continuità in quanto le linee diagonali risentono della bassa risoluzione grafica del C-64 e disegnandole non continue su un intervallo di continuità minore di otto bit (la matrice grafica sarebbe di otto bit) diventerebbero molto confuse. Inoltre bisognerebbe risolvere il problema di come disporre il byte della matrice grafica, se verticalmente oppure orizzontalmente, a seconda della pendenza della linea. Per evitare di trovarci di fronte a questi e altri problemi, che sono risolubili solo fino a un certo punto, abbiamo optato per le linee diagonali piene. Presentiamo ora le routine per disegnare le linee, e le routine per la manipolazione dei punti sullo schermo.

DrawPoint

Funzione: Disegna un punto con il colore di fondo o del tratto, oppure lo copia dal buffer di schermo.

Indirizzo: $C133

Accede a: Se disegna, cioè non copia dal buffer di schermo, accede a dispBufferOn
- bit 7 se è impostato a 1 scrive sullo schermo principale
- bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:

N	impostato a 1 per copiare un punto dal buffer di schermo, impostato a 0 per disegnare un punto sullo schermo
C	se N è impostato a 0, C impostato a 1 disegna il punto con il colore del tratto, C impostato a 0 disegna il punto con il colore di fondo
r3	coordinata x del pixel (0 - 319)
r11L	coordinata y del pixel (0 - 199)

Restituisce: r3, r11L inalterati

Distrugge: a, x, y, r5 - r6

Sinossi: Disegna un punto con il colore del tratto o di fondo, oppure copia lo stato del corrispondente bit dal buffer di schermo allo schermo principale. Queste operazioni si selezionano agendo sui bit di segno (N) e di carry (C) della parola di stato PSW del sistema. Se N è impostato a 1, allora il bit viene copiato dal buffer di schermo, cioè ripristinato sullo schermo principale. Se invece è impostato a 0 ci sono due diverse possibilità a seconda dello stato del flag di carry. Se C è impostato a 1 il punto viene disegnato con il colore del tratto (normalmente nero: il bit è impostato a 1 nella RAM di schermo), mentre se è impostato a 0 il punto viene disegnato con il colore del fondo (normalmente bianco: il bit è impostato a 0 nella RAM di schermo).

TestPoint

Funzione: Restituisce il carry con il valore del pixel indicato.

Indirizzo: $C13F

Accede a: dispBufferOn
bit 7 se è impostato a 1, il test viene effettuato sul pixel dello schermo principale
bit 6 se è impostato a 1, il test viene effettuato sul pixel del buffer di schermo
Se entrambi i bit 6 e 7 sono impostati a 1, viene restituito il risultato dall'operazione logica OR eseguita fra il pixel dello schermo principale e il pixel del buffer di schermo

Parametri: r3 coordinata x del pixel (0 - 319)
r11L coordinata y del pixel (0 - 199)

Restituisce: C 1 se il bit è impostato a 1, altrimenti 0
r3, r11L inalterati

Distrugge: a, x, y, r5 - r6

Sinossi: TestPoint restituisce il flag carry con il valore del pixel le cui coordinate sono passate in r3 e r11L. Il controllo può essere effettuato sul pixel dello schermo principale, su quello del buffer di schermo, o su entrambi, nel qual caso viene restituito il risultato dell'operazione logica OR eseguita fra i due pixel. Di solito l'istruzione jsr TestPoint viene immediatamente seguita dall'istruzione bcc o bcs in maniera che lo stato del pixel venga subito identificato.

HorizontalLine

Funzione: Disegna una linea orizzontale ripetendo la matrice grafica, o di continuità, impostata.

Indirizzo: $C118

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:
a matrice monodimensionale di continuità per disegnare la linea (non è da confondere con le matrici grafiche bidimensionali di sistema offerte da GEOS)
r3 coordinata x del limite sinistro della linea (0 - 319)
r4 coordinata x del limite destro della linea (0 - 319)
r11L coordinata y della linea (0 - 199)

Restituisce: r3, r4, r11L inalterati

Distrugge: a, x, y, r5 - r8, r11H

Sinossi: HorizontalLine disegna una linea orizzontale dalla coordinata x in r3 alla coordinata x in r4. La posizione verticale della linea è indicata dalla coordinata y contenuta in r11L. La routine visualizza o meno i pixel di una linea orizzontale attenendosi alla maschera definita dalla matrice grafica. In questo modo a ogni bit impostato a 1 dalla matrice corrisponde sullo schermo un bit a 1 e viceversa. Il byte della matrice grafica monodimensionale viene ripetutamente memorizzato nei byte della RAM riservata per lo schermo, in maniera da visualizzare la linea orizzontale. Se l'estremo sinistro della linea non corrisponde al limite del byte, il byte della matrice grafica, prima di venir scritto nella RAM dello schermo, viene opportunamente troncato. Vale lo stesso discorso per l'estremo destro della linea. Due linee disegnate una accanto all'altra impiegando la stessa matrice grafica, hanno i bit impostati a 1 e a 0 allineati verticalmente.

Nel caso si desideri visualizzare una linea orizzontale impiegando le matrici grafiche bidimensionali di sistema (8 x 8 pixel), conviene disegnare rettangoli alti un pixel, chiamando la routine Rectangle con la stessa coordinata y.

VerticalLine

Funzione: Disegna una linea verticale che dipende dalla matrice di continuità impostata.

Indirizzo: $C121

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:

a	byte della matrice grafica
r4	coordinata x della linea (0 - 319)
r3L	limite superiore della linea (0 - 199)
r3H	limite inferiore della linea (0 - 199)

Restituisce: r3L, r3H, r4 inalterati

Distrugge: a, x, y, r5L - r8L

Sinossi: Disegna una linea verticale ripetendo la matrice di continuità passata attraverso il registro a. Un bit della matrice monodimensionale impostato a 1 genera un bit a 1 sullo schermo, e viceversa. Il byte della matrice di continuità viene disposto verticalmente. Se due linee verticali sono visualizzate con la stessa matrice grafica, una accanto all'altra, hanno i bit impostati a 1 e a 0 allineati orizzontalmente. Questo allineamento si ottiene memorizzando i byte delle matrici grafiche su 8 linee di scansione indipendentemente dai limiti verticali della linea. Come avviene per le linee orizzontali, i bit della matrice grafica che oltrepasserebbero i limiti verticali della linea vengono opportunamente troncati prima di essere visualizzati.

Per disegnare una linea verticale utilizzando le matrici grafiche bidimensionali di sistema (8 x 8 pixel), conviene visualizzare un rettangolo largo un pixel chiamando la routine Rectangle con la stessa coordinata x.

InvertLine

Funzione: Inverte i bit di una linea orizzontale.

Indirizzo: $C11B

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:
r3 coordinata x dell'estremo sinistro della linea (0 - 319)
r4 coordinata x dell'estremo destro della linea (0 - 319)
r11L coordinata y della linea (0 - 199)

Restituisce: r3, r4, r11L inalterati

Distrugge: a, x, y, r5 - r8

Sinossi: La routine inverte tutti i bit che compongono una linea orizzontale. I pixel a 1 diventano a 0 e viceversa.

ImprintLine RecoverLine

Funzione: ImprintLine copia una linea orizzontale di bit dallo schermo principale nel buffer di schermo. Dal momento che questa routine non è direttamente accessibile alle applicazioni, si consiglia di chiamare ImprintRectangle indicando un rettangolo alto una linea di scansione. Rimandiamo quindi il lettore al paragrafo in cui tratteremo la routine ImprintRectangle.

RecoverLine copia una linea orizzontale dal buffer di schermo; è la funzione complementare di ImprintLine.

Indirizzo: RecoverLine $C11E

Parametri:

r3	coordinata x dell'estremo sinistro della linea (0 - 319)
r4	coordinata x dell'estremo destro della linea (0 - 319)
r11L	coordinata y della linea (0 - 199)

Restituisce: r3, r4, r11L inalterati

Distrugge: a, x, y, r5 - r8

Sinossi: ImprintLine - Copia dallo schermo principale nel buffer di schermo i bit che compongono una linea orizzontale. Per realizzare questa funzione, l'applicazione deve chiamare ImprintRectangle indicando un rettangolo alto una linea di scansione.

RecoverLine - Copia i bit che compongono una linea orizzontale dal buffer di schermo allo schermo principale. Per questa funzione, i flag contenuti nella variabile dispBufferOn vengono ignorati: i pixel vengono sempre copiati sullo schermo principale indipendentemente dai valori dei flag.

DrawLine

Funzione: Disegna, cancella, o copia dal buffer di schermo, una linea delimitata da due punti qualunque dello schermo.

Indirizzo: $C130

Accede a: DrawPoint tramite questa routine, eseguita ripetutamente, la linea viene costruita punto per punto
dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:

N	se è impostato a 1 copia i bit dal buffer di schermo, se è impostato a 0 disegna
C	se N è a 0, impostato a 1 disegna, impostato a 0 cancella
r3	x1: coordinata x del primo punto (0 - 319)
r11L	y1: coordinata y del primo punto (0 - 199)
r4	x2: coordinata x del secondo punto (0 - 319)
r11H	y2: coordinata y del secondo punto (0 - 199)

Restituisce: Registro di stato inalterato

Distrugge: a, x, y, r3 - r13

Sinossi: DrawLine attiva, disattiva o copia dal buffer di schermo i pixel compresi nella linea che unisce due punti arbitrari dello schermo. Un bit è "attivato" quando è impostato a 1, "disattivato" quando è impostato a 0.

DrawLine utilizza l'algoritmo di Breesenham per determinare quali bit sono interessati al cambiamento di stato (si consulti il libro *Fundamentals of Interactive Computer Graphics* di J. D. Foley e A. Van Dam per una spiegazione di questo algoritmo). La linea viene disegnata nello stesso modo anche scambiando fra loro i due estremi. L'algoritmo di Breesenham è stato scelto in quanto non utilizza operazioni di moltiplicazione e divisione.

DrawLine mette a disposizione diverse possibilità: copiare una linea, disegnarla (bit a 1) o cancellarla (bit a 0). I valori contenuti nei flag di segno (N) e di carry (C) della parola di stato PSW specificano quale operazione verrà effettuata. Se C è impostato a 1 la linea viene disegnata, mentre se è impostato a 0 la linea viene cancellata. Se N è impostato a 1 il flag di carry

C viene ignorato e i bit che costituiscono la linea vengono copiati dal buffer di schermo.

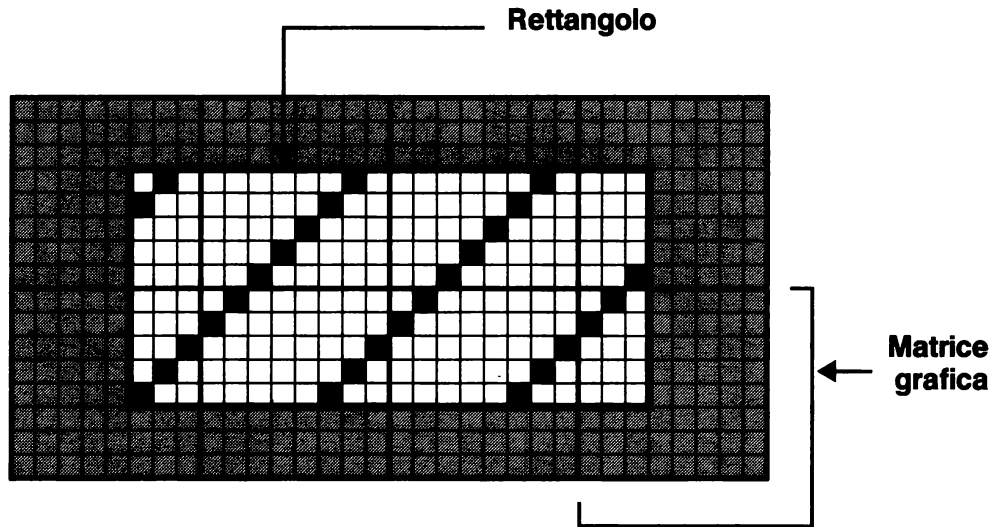
Per impostare questi flag nel modo corretto, si possono adottare alcuni "trucchi".

Usare sec e clc per impostare a 1 o a 0 il carry (C).
Usare lda #−1 per impostare a 1 il flag di segno (N).
Usare lda #0 per impostare a 0 il flag di segno (N).
Usare plp per impostare entrambi i flag secondo il valore contenuto nel primo byte dello stack.

Il disegno degli spazi pieni

Spiegheremo ora come ci si può servire di una matrice grafica per riempire gli spazi, e vedremo che concettualmente il metodo è simile a quello adottato per introdurre il disegno delle linee. In questo caso la matrice grafica non è più 1 x 8 pixel (un byte), ma 8 x 8 pixel (8 byte). Anche in questo caso la matrice grafica è sempre allineata con gli spazi carattere in maniera indipendente dall'area visualizzata. Per meglio capire cosa intendiamo per allineamento indipendente, è utile immaginare lo schermo in alta risoluzione suddiviso in tanti piccoli quadrati di 8 x 8 pixel (gli spazi carattere). Imponendo che una matrice grafica sia sempre situata all'interno di uno spazio carattere, e non a cavallo di due o più, otteniamo una gestione delle matrici grafiche molto simile alla gestione dello schermo a caratteri del C-64 quando non è selezionato alcun modo grafico. La differenza risiede nella possibilità, tipica dell'ambiente GEOS, di mascherare parte della matrice grafica prima che sia visualizzata. In questo modo, anche se i limiti della matrice grafica non coincidono con quelli dello spazio da riempire, mascherando opportunamente le parti di matrice che li supererebbero si risolve il problema. Rendendo in questo modo indipendente la posizione delle matrici grafiche da quella dello spazio da riempire, si può essere certi che due regioni adiacenti "coperte" con la stessa matrice grafica saranno identiche, con continuità lungo il bordo a entrambe comune. La dimensione di 8 x 8 pixel scelta per le matrici grafiche si presta particolarmente bene a essere adoperata in modo bit-map con il C-64, in quanto lo schermo in alta risoluzione è suddiviso, dal punto di vista operativo, in gruppi di 8 byte. Nella semplice applicazione presentata nel precedente capitolo, abbiamo utilizzato senza illustrarle nei dettagli le due routine SetPattern e i_Rectangle. SetPattern serve per selezionare la matrice grafica (pattern) di sistema prima che sia utilizzata. Tramite questa routine, GEOS riserva 8 byte alla matrice grafica di sistema selezionata. SetPattern memorizza in questi 8 byte una delle diverse matrici grafiche disponibili in ambiente GEOS. A ogni matrice grafica è assegnato un numero.

Le routine di disegno come i_Rectangle utilizzano la matrice grafica di sistema selezionata per riempire aree dello schermo. Nel caso della nostra semplice applicazione, abbiamo utilizzato SetPattern e i_Rectangle per cancellare lo schermo. GEOS manipola le aree piene come fa con le linee orizzontali e verticali. L'effetto che si ottiene riempiendo una zona dello schermo con una matrice grafica è simile a quello che si ottiene disponendo un foglio bianco su un panno disegnato e tagliandone via una parte in modo da rivelare il panno sottostante. Quando si disegna un'area piena sovrapponendola a un'altra, la matrice grafica della prima si allinea con quella della seconda.

La figura della pagina successiva rappresenta un piccolo rettangolo i cui bordi non coincidono con i bordi dei quadrati occupati dalle matrici grafiche. L'area grigia rappresenta i bit delle matrici grafiche che devono essere mascherati in maniera che i bordi del rettangolo siano quelli richiesti, indipendentemente dalla sua posizione.

L'area grigia include le parti delle matrici grafiche che vengono mascherate prima della visualizzazione

Le routine grafiche che seguono servono per disegnare forme geometriche riempite con le matrici grafiche.

SetPattern

Funzione: Seleziona la matrice grafica di sistema.

Indirizzo: $C139

Parametri: a — il numero della matrice grafica che si desidera selezionare (0 - 31)

Restituisce: curPattern — aggiorna questa variabile con l'indirizzo della matrice grafica selezionata in a

Distrugge: a

Sinossi: Imposta la matrice grafica di sistema secondo una delle 32 disponibili in ambiente GEOS. L'indirizzo della matrice grafica selezionata è contenuto nella variabile curPattern. Tutte le routine che riempiono regioni dello schermo, come Rectangle, disegnano utilizzando la matrice grafica di sistema correntemente selezionata.

Rectangle, i_Rectangle

Funzione: Disegna un rettangolo riempito con la matrice grafica correntemente selezionata.

Indirizzo: Rectangle $C124
i_Rectangle $C19F

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri: *Normali*
r2L coordinata y del lato superiore del rettangolo in linee di scansione (0 - 199)
r2H coordinata y del lato inferiore del rettangolo in linee di scansione (0 - 199)
r3 coordinata x del lato sinistro del rettangolo in pixel (0 - 319)
r4 coordinata x del lato destro del rettangolo in pixel (0 - 319)

Inline
I parametri devono apparire subito dopo l'istruzione jsr
.byte coordinata y del lato superiore del rettangolo
.byte coordinata y del lato inferiore del rettangolo
.word coordinata x del lato sinistro del rettangolo
.word coordinata x del lato destro del rettangolo

Restituisce: r2, r3, r4 inalterati

Distrugge: a, x, y, r2, r5 - r8, r11

Sinossi: Disegna un rettangolo di coordinate date riempito con la matrice grafica di sistema selezionata. Tale matrice grafica può essere cambiata chiamando SetPattern. La matrice grafica di sistema corrente è una matrice 8 x 8 pixel che viene visualizzata sullo schermo con i bordi coincidenti con quelli degli spazi carattere ottenuti supponendo di dividere lo schermo in 1000 quadrati da 8 x 8 pixel (questa suddivisione individua esattamente gli spazi carattere del modo testo del C-64). Quando il perimetro del rettangolo non corrisponde a quello dello spazio caratteri, le matrici grafiche interessate, prima di essere visualizzate, vengono parzialmente mascherate, in modo da non eccedere

i limiti del rettangolo. Si rimanda il lettore all'inizio di questo capitolo per maggiori dettagli.

Rectangle opera chiamando ripetutamente HorizontalLine in un loop, e assegnando, a ogni chiamata, il byte di continuità correntemente prelevato dalla matrice grafica di sistema selezionata. In questo modo, le applicazione possono supportare matrici grafiche con altezze maggiori (8 x 16 o 8 x 32 pixel), chiamando ripetutamente HorizontalLine in un loop.

Per incorniciare un rettangolo si deve prima chiamare Rectangle e successivamente FrameRectangle, altrimenti Rectangle cancella la cornice. C'è anche la possibilità di chiamare prima FrameRectangle, ma le coordinate da passare devono essere (x1−1, y1−1) e (x2+1, y2+1), in modo che Rectangle non possa cancellare FrameRectangle. In questo modo, però, le dimensioni del rettangolo visualizzato sono ovviamente maggiori.

FrameRectangle, i_FrameRectangle

Funzione: Visualizza le linee di una cornice utilizzando il byte della matrice di continuità impostata.

Indirizzo: FrameRectangle $C127
i_FrameRectangle $C1A2

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri: *Normali*
a byte della matrice di continuità nella forma che si utilizza per disegnare le linee orizzontali e verticali.
r2L coordinata y del lato superiore del rettangolo in linee di scansione (0 - 199)
r2H coordinata y del lato inferiore del rettangolo in linee di scansione (0 - 199)
r3 coordinata x del lato sinistro del rettangolo in pixel (0 - 319)
r4 coordinata x del lato destro del rettangolo in pixel (0 - 319)

Inline
I parametri devono apparire subito dopo l'istruzione jsr
.byte lato superiore del rettangolo
.byte lato inferiore del rettangolo
.word lato sinistro del rettangolo
.word lato destro del rettangolo
.byte matrice di continuità

Restituisce: r2 - r4 inalterati

Distrugge: a, x, y, r5 - r9, r11

Sinossi: Utilizza il byte della matrice grafica, passato attraverso a, per disegnare le linee larghe un pixel della cornice. Il byte della matrice grafica è disposto verticalmente per disegnare i lati sinistro e destro della cornice. Come per le altre routine di disegno, il byte della matrice di continuità viene allineato con

gli spazi carattere. Si rimanda all'inizio di questo capitolo per maggiori dettagli.

Per incorniciare un rettangolo pieno bisogna chiamare Rectangle prima di FrameRectangle, altrimenti Rectangle cancella la cornice. FrameRectangle disegna un rettangolo vuoto (cioè una cornice le cui linee sono larghe un pixel e sono disegnate con il colore del tratto) sopra un rettangolo pieno. Se si chiama Rectangle con gli stessi parametri con cui era stato chiamato FrameRectangle, Rectangle cancella la cornice perché GEOS considera tutte le dimensioni in modo inclusivo. C'è anche la possibilità di chiamare prima FrameRectangle, ma in questo caso le coordinate da passare devono essere (x1−1, y1−1) e (x2+1, y2+1), con la conseguenza che le dimensioni del rettangolo aumentano.

InvertRectangle

Funzione: Inverte i pixel contenuti in un rettangolo.

Indirizzo: $C12A

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:
r2L coordinata y del lato superiore del rettangolo in linee di scansione (0 - 199)
r2H coordinata y del lato inferiore del rettangolo in linee di scansione (0 - 199)
r3 coordinata x del lato sinistro del rettangolo in pixel (0 - 319)
r4 coordinata x del lato destro del rettangolo in pixel (0 - 319)

Restituisce: r2 - r4 inalterati

Distrugge: a, x, y, r5 - r8, r11L

Sinossi: I pixel contenuti nel rettangolo di dimensioni date vengono invertiti. I valori logici 1 diventano 0 e viceversa. L'effetto è quello di cambiare il colore del tratto in colore di fondo e il colore di fondo in colore del tratto. Questa routine può essere utile per segnalare all'utente l'avvenuta selezione di un oggetto.
InvertRectangle opera chiamando InvertLine all'interno di un loop.

RecoverRectangle, i_RecoverRectangle

Funzione: Copia un rettangolo dal buffer di schermo.

Indirizzo: RecoverRectangle $C12D
i_RecoverRectangle $C1A5

Accede a: dispBufferOn ignorato

Parametri: *Normali*

r2L coordinata y del lato superiore del rettangolo in linee di scansione (0 - 199)
r2H coordinata y del lato inferiore del rettangolo in linee di scansione (0 - 199)
r3 coordinata x del lato sinistro del rettangolo in pixel (0 - 319)
r4 coordinata x del lato destro del rettangolo in pixel (0 - 319)

Inline

I parametri devono apparire subito dopo l'istruzione jsr
.byte coordinata y del lato superiore del rettangolo
.byte coordinata y del lato inferiore del rettangolo
.word coordinata x del lato sinistro del rettangolo
.word coordinata x del lato destro del rettangolo

Restituisce: r2 - r4 inalterati

Distrugge: a, x, y, r5 - r8, r11

Sinossi: I pixel contenuti nel rettangolo delle dimensioni date vengono copiati dal buffer sullo schermo principale in un rettangolo delle stesse dimensioni e identica posizione. Il disegno dello stesso rettangolo presente sullo schermo principale viene perso. È bene ricordare che il disegno nel buffer di schermo dev'essere già presente prima della chiamata a questa routine, in modo che sia effettivamente presente qualcosa da copiare.

ImprintRectangle, i_ImprintRectangle

Funzione: Copia i bit contenuti in un rettangolo di dimensioni date dallo schermo principale nel buffer di schermo. È la funzione complementare di RecoverRectangle.

Indirizzo: ImprintRectangle $C250
i_ImprintRectangle $C253

Accede a: dispBufferOn ignorato

Parametri: *Normali*

r2L coordinata y del lato superiore del rettangolo in linee di scansione (0 - 199)
r2H coordinata y del lato inferiore del rettangolo in linee di scansione (0 - 199)
r3 coordinata x del lato sinistro del rettangolo in pixel (0 - 319)
r4 coordinata x del lato destro del rettangolo in pixel (0 - 319)

Inline

I parametri devono apparire subito dopo l'istruzione jsr
.byte coordinata y del lato superiore del rettangolo
.byte coordinata y del lato inferiore del rettangolo
.word coordinata x del lato sinistro del rettangolo
.word coordinata x del lato destro del rettangolo

Restituisce: r2 - r4 inalterati

Distrugge: a, x, y, r5 - r8, r11L

Sinossi: ImprintRectangle copia un rettangolo dallo schermo principale al buffer di schermo nella stessa posizione. Chiamando RecoverRectangle con gli stessi parametri si ottiene la copia nella direzione opposta, cioè dal buffer di schermo allo schermo principale.

La grafica in modo bit-map

Può capitare di voler realizzare figure complicate, che non possono essere ottenute semplicemente con un insieme di linee, cornici e rettangoli pieni. In questi casi è utile gestire la figura come una zona di schermo bit-map in alta risoluzione, e memorizzarla come un gruppo di dati grafici. L'unico problema per gestire così le figure è che in modo bit-map occupano molto spazio di memoria. Quindi è necessario creare alcuni algoritmi in grado di minimizzare il numero di byte che definiscono una figura in bit-map.

In generale, i dati grafici che definiscono le figure si adattano perfettamente a essere compattati con metodi che minimizzano le ripetizioni (Run-Lenght Encoding Method). Con questi metodi, i dati in modo bit-map vengono considerati un'unica lunga stringa di byte. Prima di proseguire, è bene precisare che il lettore non deve confondere l'argomento che stiamo trattando con le specifiche imposte dal C-64 per visualizzare i dati in modo bit-map; la compattazione dei dati serve unicamente per conservare le schermate in uno spazio di memoria minore, ma dev'essere disabilitata quando si richiede la visualizzazione dei dati. Quindi dire che i metodi di compattazione considerano i dati grafici come una lunga stringa di byte, non significa dire che è cambiato il modo di visualizzazione delle schermate imposto dal C-64. Si tratta di due argomenti completamente diversi. La compattazione e la successiva espansione dei dati sono operazioni indipendenti dalla loro visualizzazione. A livello teorico si potrebbe pensare di compattare anche i dati che compongono un file di testo ASCII, ma in realtà non è conveniente perché il metodo di compattazione che si utilizza è efficace solo se nel gruppo di dati esistono molte ripetizioni sequenziali di singoli byte o di gruppi di byte. E questo si verifica più facilmente quando i dati definiscono le schermate in modo bit-map.

Si verifica spesso, nei gruppi di dati grafici che definiscono una figura in modo bit-map, che il valore di un byte sia ripetuto lungo il gruppo di dati molte volte prima d'incontrare un nuovo valore. Questa sequenza di byte uguali può essere rappresentata in due soli byte, il primo dei quali contiene il numero di ripetizioni e il secondo il valore da ripetere. La compattazione dei dati è un argomento estremamente importante e molti testi di informatica lo trattano in modo esauriente. In questa sede ci limiteremo a parlare delle tecniche principali che si usano con GEOS.

GEOS rende disponibili tre distinti metodi di compattazione per memorizzare i dati grafici in modo bit-map. I dati di definizione delle figure in ambiente GEOS che sono compattati secondo uno dei tre metodi, sono definiti "mappe grafiche" All'interno della stessa mappa grafica possono essere impiegati anche tutti e tre i metodi di compattazione, scegliendo di volta in volta quello che può fornire l'ottimizzazione migliore. Questa opportunità permette al programma di compattazione di gestire i tre metodi in modo conforme alla struttura dei dati che compongono la mappa grafica. In realtà sono solo due i veri metodi di compattazione, mentre il terzo si limita a indicare il numero dei singoli byte che seguono (ognuno dei quali deve apparire una sola volta).

Il primo byte di una mappa grafica è sempre un byte di conteggio, o contatore. All'interno di questo byte risiedono due importanti informazioni: il formato della compattazione e il numero di byte interessati. Il byte contatore e i dati a esso associati costituiscono un gruppo grafico compattato. Una serie di gruppi grafici compattati rappresenta una mappa grafica.

I formati di compattazione

Gli algoritmi di compattazione e di espansione considerano la mappa grafica suddivisa in linee di scansione orizzontali, e non come gruppi di 8 byte costituenti caratteri, come avviene durante la visualizzazione in modo bit-map con il C-64. Ci teniamo a ribadire che la compattazione e la visualizzazione sono due argomenti diversi e indipendenti l'uno dall'altro.

Quando una mappa grafica non è compattata, qualsiasi byte in posizione N ha alla sua destra, sulla stessa linea di scansione, il byte in posizione N+1 (tranne quando il byte in posizione N è l'ultimo della linea di scansione e il byte in posizione N+1 è il primo della linea di scansione successiva). Questa organizzazione dei dati grafici è molto diversa dal modo in cui il C-64 gestisce i dati grafici in memoria. La ragione per cui in ambiente GEOS le schermate non compattate, prima di essere visualizzate, vengono memorizzate in questo modo, si spiega tenendo presente che la compattazione dei byte è molto più efficace se viene effettuata secondo le linee orizzontali dello schermo, se cioè agisce sui byte di una stessa linea di scansione. Quando GEOS deve visualizzare una mappa grafica compattata, compie due operazioni. Per prima cosa analizza il formato della compattazione e ricava i dati grafici in forma non compatta. Quindi li memorizza, riordinandoli completamente secondo le esigenze del modo bit-map del C-64: il secondo byte ottenuto dall'espansione, per esempio, dev'essere memorizzato otto locazioni di memoria dopo la locazione in cui è stato memorizzato il primo, perché appaia sulla stessa linea di scansione. Con questo metodo la visualizzazione delle figure è più lenta perché, oltre a espandere, GEOS deve anche riordinare i dati, ma dal momento che l'accesso al disco è molto più lento di qualsiasi altra operazione, compattare i dati e diminuire così il numero dei blocchi presenti sul disco rappresenta comunque un buon risparmio di tempo. Analizziamo ora i diversi formati di compattazione.

Ogni gruppo grafico compattato inizia con un byte di conteggio (Count Byte). Il suo valore determina anche il formato della compattazione. Per brevità, chiamiamo il valore del byte di conteggio COUNT. Se COUNT è compreso fra 0 e 127, cioè il bit 7 è sempre a 0, il gruppo grafico è stato compattato con il primo formato. Un numero compreso tra 128 e 220 indica il secondo formato, mentre un numero compreso tra 221 e 255 indica il terzo. Nel primo formato, il secondo byte è ripetuto un numero di volte pari al valore specificato da COUNT. Se per esempio il byte di conteggio contiene il valore 100 e il secondo byte il valore 0, quest'ultimo viene ripetuto 100 volte nella

schermata grafica da visualizzare in bit-map. Se COUNT è compreso fra 128 e 220 (secondo formato), i (COUNT – 128) byte che lo seguono sono riportati nella schermata grafica sequenzialmente, e ognuno compare una sola volta. Per esempio, se COUNT è uguale a 128 + 35 = 163, i 35 byte che seguono COUNT devono essere trascritti sequenzialmente uno per volta nella schermata grafica. Se COUNT è compreso fra 221 e 255 (terzo formato), i byte che seguono sono da considerare in formato BIGCOUNT, un metodo di compattazione che merita qualche parola in più.

BIGCOUNT è stato creato per comandare la ripetizione di una matrice grafica composta da diversi byte per un certo numero di volte. Spieghiamo questo formato direttamente con un esempio. Supponiamo di avere una matrice grafica composta da quattro byte (xxxy), e di volerla ripetere cinque volte:

xxxy xxxy xxxy xxxy xxxy

Ogni matrice grafica è composta dai quattro byte xxxy. Per compattare questo gruppo di dati con il formato BIGCOUNT, si devono prima di tutto descrivere i quattro byte che compongono la matrice grafica utilizzando uno degli altri formati che abbiamo già analizzato. xxxy può essere descritta con il secondo formato: 132,x,x,x,y. In questo caso la chiave di interpretazione è: ripeti i prossimi quattro byte una volta ciascuno, sequenzialmente. Oppure la matrice grafica può essere compattata con il primo formato: 3,x,1,y. La chiave di interpretazione con il primo formato è: ripeti x tre volte e y una volta.

Quando la matrice grafica è stata interamente descritta con il primo o il secondo formato, GEOS deve sapere quanti byte occupa la descrizione della matrice grafica e quante volte dev'essere interamente ripetuta. Il primo byte del gruppo grafico compattato con il formato BIGCOUNT è il byte di conteggio il cui valore COUNT identifica il terzo formato e il numero di byte che compongono la singola matrice grafica (COUNT – 220). Il secondo byte contiene il numero di volte che GEOS deve ripetere la matrice grafica per intero. In conclusione il gruppo grafico compattato dell'esempio che stiamo facendo deve apparire come la seguente linea di byte:

224,5,3,x,1,y oppure 224,5,132,x,x,x,y

Si può notare che descrivendo la matrice grafica con il primo formato si risparmia un byte. In ogni caso, la ripetizione della matrice per cinque volte occuperebbe 20 byte, mentre compattata con il sistema che abbiamo spiegato occupa 6 byte con un risparmio di 14 byte. È chiaro quindi che la compattazione, usata nel modo più efficiente, riduce notevolmente il numero di byte che definiscono un disegno in modo bit-map. Segue una tavola riassuntiva dei tre formati di compattazione.

Formati di compattazione

Count	Formato	Descrizione
0 - 127	COUNT,DATO	Ripete COUNT volte il dato DATO
128 - 220	COUNT,DATO1,DATO2 ...	Ripete i (COUNT - 128) dati che seguono una volta per uno
221 - 255	COUNT,BIGCOUNT,MATRICE	(COUNT - 220) = numero di byte che compongono la matrice grafica, o PATTERN. Questo numero non comprende BIGCOUNT. BIGCOUNT = numero che indica quante volte la matrice grafica dev'essere ripetuta. MATRICE = insieme di byte (nel primo o nel secondo formato) che definisce la matrice grafica da ripetere

Riassumendo, la mappa grafica è un insieme di gruppi grafici compattati, che utilizzano almeno uno dei tre formati di compattazione descritti. Un gruppo grafico compattato inizia con un byte che ne specifica il formato seguito da una serie di byte il cui significato dipende dal tipo di formato indicato. Dopo l'espansione, i byte del disegno sono riordinati da linee di scansione a spazi carattere di 8 byte, in modo da renderne possibile la visualizzazione in modo bit-map da parte del C-64.

BitmapUp, i_BitmapUp

Funzione: Visualizza un rettangolo in modo bit-map prelevandone i dati da una mappa grafica compattata.

Indirizzi:
BitmapUp $C142
i_BitmapUp $C1AB

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri: *Normali*
r0 puntatore ai dati della mappa grafica compattata
r1L coordinata x in byte del lato sinistro del rettangolo (0 - 39)
r1H coordinata y in linee di scansione del lato superiore del rettangolo (0 - 199)
r2L larghezza in byte del rettangolo (0 - 39)
r2H altezza in pixel del rettangolo (0 - 199)

Inline
i parametri devono apparire subito dopo l'istruzione jsr
.word puntatore ai dati della mappa grafica compattata
.byte coordinata x in byte del lato sinistro del rettangolo (0 - 39)
.byte coordinata y in linee di scansione del lato superiore del rettangolo (0 - 199)
.byte larghezza in byte del rettangolo (0 - 39)
.byte altezza in pixel del rettangolo (0 - 199)

Restituisce: r11L inalterato

Distrugge: a, x, y, r0 - r9L

Sinossi: BitmapUp visualizza una mappa grafica compattata sullo schermo. BitmapUp espande i dati e colloca il disegno sullo schermo in una posizione che dipende dalle coordinate x e y dei lati sinistro e superiore e dalle dimensioni del rettangolo da visualizzare. Abbiamo già illustrato ampiamente i tre formati di compattazione. BitmapUp non controlla se la posizione e le dimensioni del rettangolo sono corrette. Per ottenere la visualizzazione parziale di una mappa grafica si deve utilizzare la routine BitmapClip, che ora descriveremo.

Talvolta può rendersi utile visualizzare una mappa grafica solo in parte. La routine BitmapClip permette al programmatore di visualizzare all'interno di una finestra una parte della mappa grafica. Solo la porzione della mappa grafica che appare all'interno della finestra sarà visibile. Il resto del disegno è mascherato e non viene visualizzato. C'è anche la possibilità di specificare quale parte della mappa grafica dev'essere visualizzata all'interno della finestra aperta sullo schermo. In un certo senso si può pensare di far scorrere la mappa "sotto lo schermo", in modo che l'utente ne veda solo quella parte che viene "rivelata" dalla finestra aperta.

A questo scopo, il programmatore deve innanzitutto specificare le coordinate della finestra e le deve passare attraverso i registri r1 e r2, come vedremo fra poco. Quindi, nei registri r11L e r11H, deve decidere quanti byte GEOS deve saltare prima di iniziare a visualizzare i dati della mappa grafica. Memorizzando il valore 0 in r11L, si fa corrispondere il lato sinistro della mappa grafica con il lato sinistro della finestra. Un valore pari a 10 in r11L significa invece che prima di iniziare a visualizzare le linee di scansione della mappa grafica, i primi 10 byte di ogni linea di scansione devono essere mascherati in modo che il lato sinistro della finestra corrisponda all'undicesimo byte di ogni linea di scansione.

Se la mappa grafica, a partire dalla coordinata orizzontale specificata da r11L, è più larga della finestra in cui deve apparire, una parte della fine di ogni linea di scansione della schermata dev'essere mascherata. Il numero di byte in eccesso dev'essere specificato in r11H e rappresenta la seconda coordinata orizzontale della parte di mappa grafica da visualizzare. Quando questi parametri sono stati specificati, la mappa grafica si trova a essere divisa orizzontalmente in tre parti: la prima parte è il numero di byte da saltare, prima di iniziare a visualizzare (memorizzato in r11L) la seconda parte corrisponde alla larghezza in byte della finestra e la terza parte è costituita dai rimanenti byte che non devono essere visualizzati (il cui numero è memorizzato in r11H).

La collocazione verticale della mappa grafica parziale si ottiene in modo analogo. Il numero di righe di scansione che devono essere saltate prima dell'inizio è contenuto nella word r12: variando il valore contenuto in r12 si ottiene lo scroll verticale della mappa grafica all'interno della finestra. Ecco le specifiche che caratterizzano la routine BitmapClip.

BitmapClip

Funzione: Visualizza all'interno di una finestra aperta sullo schermo un'area parziale di una mappa grafica.

Indirizzo: $C2AA

Accede a: dispBufferOn
- bit 7 se è impostato a 1 scrive sullo schermo principale
- bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:

r0	puntatore all'inizio della mappa grafica in memoria
r1L	coordinata x del lato sinistro in byte della finestra dove viene visualizzata la mappa grafica (0 - 39)
r1H	coordinata y del lato superiore in pixel della finestra (0 - 199)
r2L	larghezza in byte della finestra (0 - 39)
r2H	altezza in pixel della finestra (0 - 199)
r11L	numero di byte dall'inizio di ogni linea di scansione della mappa grafica da saltare prima di iniziare la visualizzazione all'interno della finestra
r11H	numero di byte rimanenti, a partire dalla fine della linea di scansione della mappa grafica, che non vengono visualizzati dopo che è stata disegnata la parte che riempie la finestra
r12	questa word specifica il numero di linee di scansione, a partire dalla prima in alto della mappa grafica, che devono essere saltate prima di iniziare la visualizzazione del disegno all'interno della finestra

Restituisce: Niente

Distrugge: a, x, y, r0 - r12

Sinossi: La routine BitmapClip si utilizza per visualizzare una parte di mappa grafica all'interno di una finestra definita dal programmatore. La finestra può trovarsi in qualunque posizione sullo schermo e può essere di dimensioni arbitrarie: rappresenta, di solito, uno spazio di lavoro dell'applicazione. r1L e r1H, insieme a r2L e r2H, definiscono le dimensioni della finestra al cui interno deve comparire la parte di mappa che si desidera visualizzare. r1L e r1H contengono le coordinate della posizione dell'angolo sinistro superiore della finestra, mentre r2L ed r2H ne definiscono le dimensioni in larghezza e in

altezza. r0 deve contenere l'indirizzo in memoria ove inizia la mappa grafica da visualizzare.

Dal momento che la larghezza della mappa può essere maggiore della larghezza della finestra, il programmatore deve specificare quale parte della mappa grafica deve apparire all'interno della finestra. Questa informazione viene passata attraverso r11 ed r12. Ogni volta che BitmapClip "legge" una linea di scansione della mappa grafica, controlla il valore contenuto in r11L (che rappresenta il numero di byte da saltare all'inizio di ogni linea di scansione prima di iniziare a visualizzarla all'interno della finestra). r11H contiene il numero di byte in eccesso sulla linea di scansione. Attenzione: la larghezza della mappa grafica dev'essere uguale alla somma dei valori contenuti in r11L (parte sinistra non visualizzata), r11H (parte destra non visualizzata) e r2L (larghezza della finestra). r12 contiene il numero di linee di scansione, a partire dalla prima in alto della mappa grafica, che devono essere saltate prima di iniziare a visualizzare il disegno all'interno della finestra.

Qualche volta può essere necessario dover visualizzare in parte una mappa grafica molto grande, senza che questa possa essere contenuta completamente in memoria. La routine BitOtherClip permette di specificare una routine di input che restituisce in a il successivo byte, dopo l'analisi dell'eventuale compattazione. BitOtherClip continua a chiamare la routine di input, di solito ReadByte, fino a quando non ha completamente ricevuto un gruppo grafico compattato. Quindi lo espande e lo visualizza all'interno di una finestra aperta sullo schermo, le cui dimensioni sono già state impostate.

BitOtherClip

Funzione: Permette al programmatore di costruirsi una routine di input da utilizzare con BitmapClip.

Indirizzo: $C2C5

Accede a: dispBufferOn
- bit 7 se è impostato a 1 scrive sullo schermo principale
- bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri:
- r0 puntatore a un buffer in memoria da 134 byte
- r1L coordinata x in byte del lato sinistro della finestra all'interno della quale deve apparire il disegno in bit-map (0 - 39)
- r1H coordinata y in pixel del lato superiore della finestra (0 - 199)
- r2L larghezza in byte della finestra da aprire sullo schermo (0 - 39)
- r2H altezza in pixel della finestra da aprire sullo schermo (0 - 199)
- r11L numero di byte da saltare dall'inizio della linea di scansione prima di iniziarne la visualizzazione all'interno della finestra
- r11H numero di byte rimanenti, a partire dalla fine della linea di scansione della mappa grafica, che non sono visualizzati dopo che è stata disegnata la parte che riempie in larghezza la finestra
- r12 questa word indica il numero di linee di scansione, a partire dalla prima in alto della mappa grafica, che devono essere saltate prima di iniziare la visualizzazione del disegno
- r13 indirizzo della routine di input che restituisce il successivo byte della mappa grafica senza effettuare l'espansione
- r14 indirizzo della routine che BitOtherClip esegue quando ha terminato di gestire l'ultimo gruppo grafico e si prepara ad accettare il successivo. In virtù di alcuni miglioramenti realizzati in BitmapClip, questa routine non deve far altro che aggiornare nuovamente r0 con l'indirizzo del buffer da 134 byte

Restituisce: Niente

Distrugge: a, x, y, r0 - r14

Sinossi: Qualche volta il programmatore può incontrare la necessità di visualizzare parte di una schermata che non può essere interamente contenuta in

memoria. Le mappe grafiche infatti possono essere anche molto grandi. BitOtherClip permette al programmatore di costruire una routine di input che restituisce in a il successivo byte compattato (in uno dei tre formati che abbiamo già introdotto). Questa routine non deve alterare i registri r0 - r13, e deve restituire un byte alla volta proveniente da una mappa grafica memorizzata su disco. Per compiere questa operazione deve salvare i registri, chiamare la routine ReadByte (ReadByte restituisce il byte della mappa grafica in a; per maggiori dettagli sulla routine ReadByte consultare il capitolo dedicato alla gestione dei file), e infine deve ripristinare i registri.

BitOtherClip continua a chiamare la routine puntata da r13 fino a quando i byte prelevati non compongono un gruppo grafico compattato. I byte ricevuti vengono memorizzati nel buffer specificato da r0. BitOtherClip espande il gruppo grafico contenuto nel buffer e lo visualizza sullo schermo. Quando la visualizzazione del gruppo grafico è finita, viene eseguita la routine il cui indirizzo è contenuto in r14. Questa routine non è particolarmente utile e dovrebbe semplicemente riaggiornare r0 con l'indirizzo del buffer da 134 byte in maniera tale che BitOtherClip possa prelevare il gruppo grafico successivo e visualizzarlo. Il buffer destinato a questa routine è lungo 134 byte, cioè la massima ampiezza possibile per un gruppo grafico.

Le routine GraphicsString

Durante la progettazione di GEOS ci siamo accorti che le sue routine grafiche si prestano molto bene alla realizzazione della veste grafica iniziale di un'applicazione. Lo schermo di un'applicazione si può ottenere chiamando una dopo l'altra le routine grafiche di GEOS. L'unico inconveniente è che per ottenere un risultato di qualità, il numero delle routine da eseguire è abbastanza elevato. Così abbiamo pensato di offrire un piccolo aiuto al programmatore, dotando GEOS della routine GraphicsString, che consente di creare una stringa di dati contenente tutte le operazioni grafiche da effettuare. A ogni routine è assegnato un numero che la identifica. La stringa grafica deve contenere i numeri delle routine da chiamare, seguiti dai rispettivi parametri. Tramite questo accorgimento si possono lasciare liberi tutti i byte normalmente utilizzati per aggiornare i registri con i parametri e per contenere le istruzioni jsr.

GraphicsString ha alcune caratteristiche degne di nota. La più interessante è che mantiene in memoria l'ultima posizione della simbolica "penna" usata per disegnare. Molti comandi grafici si riferiscono alla posizione della penna sullo schermo, quando iniziano a disegnare. Per esempio, il comando LINETO disegna una linea dalla corrente posizione della penna alle coordinate x e y che seguono immediatamente il comando. La posizione che individua la fine della linea, ovvero quella di coordinate x e y, diventa la nuova posizione della penna sullo schermo.

L'unica limitazione imposta da GraphicsString consiste nell'impossibilità di disegnare linee discontinue. Tutte le linee disegnate attraverso GraphicsString, come ad esempio i bordi dei rettangoli, sono continue.

I parametri di cui abbiamo parlato vengono riassunti nella scheda che segue.

GraphicsString, i_GraphicsString

Funzione: Esegue i comandi grafici indicati nella stringa grafica.

Indirizzo: GraphicsString $C136
i_GraphicsString $C1A8

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo

Parametri: *Normali*
r0 puntatore all'inizio della stringa grafica in memoria

Inline
I dati della stringa grafica devono apparire subito dopo l'istruzione jsr

Restituisce: Niente

Distrugge: a, x, y, r0 - r13

Sinossi: GraphicsString esegue i comandi contenuti all'interno di una stringa grafica. Sono previsti 11 comandi grafici numerati dal comando 0 al comando 10. Ogni comando grafico dev'essere seguito dai gruppi di coordinate che lo caratterizzano. Le coordinate orizzontali occupano sempre due byte (0 - 319), mentre le coordinate verticali occupano un solo byte (0 - 199). I comandi sono illustrati nella tavola della pagina successiva.

Comandi riconosciuti da GraphicsString

Comando	Codice	Dati	Descrizione
NULL	0	-	Fine della stringa grafica
MOVEPENTO	1	.word x .byte y	Posiziona la "penna" alle coordinate assolute x e y
LINETO	2	.word x .byte y	Disegna una linea dalla corrente posizione della penna al punto di coordinate x e y, il quale diventa, a disegno avvenuto, la nuova posizione della penna
RECTANGLETO	3	.word x .byte y	Disegna un rettangolo dalla posizione corrente della penna alle coordinate x e y del vertice opposto lungo la diagonale; x e y diventano le coordinate della nuova posizione della penna
PENFILL	4	N/A	Questo comando non e' attualmente disponibile. Inserendolo nella tavola non viene eseguita alcuna operazione
NEWPATTERN	5	.byte patternNu	Assume come matrice grafica di sistema quella il cui numero e' patternNu
ESC_PUTSTRING	6	.word xpos .byte ypos .byte stringa	Comando per fare in modo che GEOS consideri i rimanenti dati come parametri inline della routine i_PutString
FRAME_RECTO	7	.word x .byte y	Crea una cornice dalla posizione corrente della penna al punto individuato dalle coordinate x e y del vertice opposto lungo la diagonale; x e y diventano le coordinate della nuova posizione della penna
PENXDELTA	8	.word dx	Aggiunge il valore dx alla coordinata orizzontale della penna
PENYDELTA	9	.byte dy	Aggiunge il valore dy alla coordinata verticale della penna
PENXYDELTA	10	.word dx .byte dy	Aggiunge i valori dx e dy rispettivamente alla coordinata orizzontale e a quella verticale della penna

Esempio: Disegna un rettangolo con la matrice grafica numero 0 dal punto di coordinate x e y al punto di coordinate x' e y1, situato lungo la diagonale.

```
jsr      i_GraphicsString
.byte    NEWPATTERN, 0
.byte    MOVEPENTO
.word    x
.byte    y
.byte    RECTANGLETO
.word    x1
.byte    y1
.byte    NULL
```

GetScanLine

Funzione: Restituisce l'indirizzo di una linea di scansione dello schermo prescelto.

Indirizzo: $C13C

Parametri: x contiene la coordinata y della linea di scansione

Restituisce: x inalterato
r5 e r6 indirizzi del primo byte della linea di scansione restituiti secondo il valore contenuto in dispBufferOn:

bit 7	bit 6	restituisce
1	1	r5 = schermo principale; r6 = buffer di schermo
0	1	r5, r6 = buffer di schermo
1	0	r5, r6 = schermo principale
0	0	errore: r5, r6 = indirizzo del centro dello schermo

Distrugge: a

Sinossi: GetScanLine restituisce l'indirizzo del primo byte della linea di scansione la cui coordinata y viene passata in x. La routine memorizza gli indirizzi del primo byte della linea nei registri r5 e r6, secondo il valore contenuto in dispBufferOn. In questo modo l'applicazione, avendo già predisposto dispBufferOn, può impiegare entrambi i registri senza preoccuparsi che puntino al buffer di schermo o allo schermo principale. Eventualmente memorizza il dato due volte nella stessa locazione. Per esempio, l'applicazione che ha già impostato dispBufferOn ed eseguito GetScanLine può agire come segue:

```
ldy     xpos      ;indice del byte all'interno della linea
lda     grByte    ;valore grafico da memorizzarvi
sta     (r5), y   ;eventualmente memorizza in entrambi gli schermi
sta     (r6), y
```

5 I TESTI IN AMBIENTE GEOS

Una delle migliori qualità di GEOS è la capacità di gestire in modo autonomo i testi, offrendo numerose alternative per quanto riguarda fonti carattere, stili e corpi. Queste caratteristiche sono disponibili per qualsiasi tipo d'applicazione. Naturalmente esistono applicazioni che hanno bisogno di queste potenzialità solo in parte, e altre, come per esempio un programma di desktop publishing, che utilizzano in maniera completa tutto quello che GEOS ha da offrire. Per soddisfare entrambe le esigenze, abbiamo suddiviso le routine di manipolazione dei testi in due livelli. Al livello più semplice GEOS mette a disposizione alcuni comandi per gestire frasi in input e in output. A questo livello la maggior parte del lavoro, come la spaziatura proporzionale e il cambio dello stile, viene eseguito da GEOS per intere frasi, liberando così il programmatore dal compito di creare un'articolata e complessa struttura di gestione dei caratteri. Sia l'input di testi digitati dall'utente, con la relativa generazione di eco sullo schermo, sia l'output su schermo, sono procedure interamente gestite dal Kernel di GEOS. L'unica operazione che per adesso GEOS non è in grado di compiere all'interno di una frase è la selezione di un'altra fonte carattere, o un cambio di corpo, perché la gestione delle fonti carattere può richiedere grandi spazi di memoria ed è l'applicazione – non il sistema – che deve provvedere ad allocarli opportunamente.

GEOS comunque possiede alcune routine di facile uso, utili alla realizzazione di strutture in grado di manipolare le fonti carattere. La selezione di una nuova fonte carattere all'interno di una linea richiede che l'applicazione lasci libera una parte della memoria, carichi la fonte carattere e la selezioni come fonte di sistema.

Prima di tutto tratteremo le routine di più alto livello. Illustreremo, con un semplice esempio, la gestione di un testo nel quale la maggior parte del lavoro sarà compiuto da GEOS. Nel prossimo capitolo approfondiremo, invece, le routine di livello inferiore per la gestione dei singoli caratteri e delle fonti carattere. Il programmatore le potrà utilizzare per creare strutture di gestione dei testi anche molto complesse e articolate,

ma in questi casi il lavoro che dovrà essere svolto dall'applicazione, come vedremo, sarà molto maggiore.

Manipolazione semplificata delle stringhe

La tecnica più semplice che GEOS è in grado di offrire per realizzare l'input e l'output di testi consiste nell'impiego delle routine PutString e GetString. PutString visualizza una stringa di testo indicata dall'applicazione. GetString accetta i caratteri digitati dall'utente, ne visualizza l'eco sullo schermo, e li alloca in una stringa di testo a terminazione nulla, all'interno di un buffer opportunamente assegnato. In genere queste due routine hanno il compito di visualizzare un messaggio all'utente (PutString) e successivamente di attendere la risposta (GetString). Vediamo come si utilizza PutString.

La maggior parte delle stringhe utilizzate da GEOS devono terminare con il carattere nullo, che in ASCII ha il codice 0. Questo significa che la fine di una stringa è identificata da un byte di valore 0. Ecco un esempio di una stringa a terminazione nulla:

```
TextLabel:          .byte  "Questa e' una stringa a terminazione nulla.",0
```

Il nostro compilatore Assembly riconosce i caratteri racchiusi tra una coppia di doppie virgolette come codici ASCII. Quindi i caratteri ASCII per la stringa "Questa e' una stringa a terminazione nulla." sono memorizzati sequenzialmente. L'ultimo carattere è il carattere nullo (0), che non viene visualizzato ed è interpretato come terminatore della stringa. Una stringa di caratteri strutturata in questo modo si dice "a terminazione nulla". PutString visualizza la stringa a partire da una particolare posizione sullo schermo che dev'essere specificata dall'applicazione. La chiamata a PutString per visualizzare la stringa si realizza con le seguenti istruzioni:

```
LoadW    r0, TextLabel       ;r0 punta il primo byte della stringa in memoria
LoadB    r1, YPOSCONSTANT    ;coordinata y della posizione in cui visualizzare
                             ;la stringa (0 - 199)
LoadW    r11, XPOSCONSTANT   ;coordinata x della posizione in cui visualizzare
                             ;la stringa (0 - 319)
jsr      PutString
```

nelle quali YPOSCONSTANT e XPOSCONSTANT sono le coordinate che individuano la posizione sullo schermo dell'angolo sinistro in basso del primo carattere della stringa. È disponibile anche la forma inline di passaggio dei parametri: i_PutString.

La chiamata inline è la seguente:

```
jsr     i_PutString       ;chiamata alla routine
.word   XPOSCONSTANT      ;coordinata x (0 - 319)
.byte   YPOSCONSTANT      ;coordinata y (0 - 199)
.byte   "Questa e' una stringa a terminazione nulla",0    ;la stringa da visualizzare
...                       ;al ritorno dalla routine il program counter riprende da qui
```

Il vettore di errore

Cosa accade se si verifica un errore? Supponiamo che la stringa che si desidera visualizzare utilizzando PutString sia sufficientemente larga da oltrepassare il limite destro dello schermo. Quando PutString visualizza la stringa e questa raggiunge il limite destro dello schermo, GEOS controlla se è presente un indirizzo nel vettore stringFaultVec. Questo vettore è generalmente impostato a 0 da GEOS durante l'inizializzazione del sistema. Se il valore in esso contenuto è ancora 0 quando PutString indica che la stringa oltrepassa il margine destro dello schermo, allora non verrà visualizzato nessun altro carattere della stringa. Se invece l'applicazione ha memorizzato in stringFaultVec l'indirizzo di una routine, GEOS la esegue, passando in a il valore ASCII del carattere la cui visualizzazione oltrepasserebbe il margine destro, in r11L la coordinata x e in r11H la coordinata y del carattere.

I margini utilizzati da PutString devono essere memorizzati nelle variabili leftMargin e rightMargin. La routine attivata attraverso il vettore stringFaultVec, controlla le coordinate x e y per individuare quale margine è stato oltrepassato. Di solito ha il compito di visualizzare a capo i caratteri che superano il margine destro. Effettua questa operazione tramite la routine PutChar che analizzeremo nel prossimo capitolo. Sempre nel prossimo capitolo vedremo come operare una perfetta impaginazione del testo entro i margini definiti.

Cambiamenti di stile all'interno delle stringhe

PutString è in grado inoltre di gestire i caratteri che controllano i cambiamenti di stile all'interno delle stringhe. Questi caratteri sono speciali valori ASCII che normalmente non sono stampabili, come i caratteri di controllo ottenuti con il tasto control (CTRL). Il primo carattere ASCII visualizzabile è il 32, che indica lo spazio. GEOS utilizza i valori ASCII minori di 32, destinandoli a comandi che modificano lo stile della stampa, per esempio da nero (bold) a nero corsivo (bold italic). Noi chiamiamo questi caratteri di controllo "caratteri escape", in quanto obbligano momentaneamente PutString a lasciare l'operazione in corso e a cambiare lo stile. In seguito, PutString continuerà a visualizzare in nero corsivo i successivi caratteri.

Quello che accade si può descrivere come segue: PutString sta visualizzando una stringa sullo schermo utilizzando lo stile nero, quando incontra tra i caratteri della stringa il carattere escape 25 (decimale), che all'interno di una stringa da visualizzare indica lo stile corsivo. Allora interrompe per un attimo la visualizzazione dei caratteri ed effettua il cambiamento di stile. I successivi caratteri saranno visualizzati in nero corsivo. In altre parole, gli effetti dei due comandi escape si sommano: un comando di corsivo *non cancella* il precedente comando di nero.

Quando PutString incontra un carattere escape, memorizza nella variabile GEOS currentMode il valore del nuovo stile. In currentMode ogni stile è associato a un bit. Quando si incontra un carattere escape, il corrispondente bit in currentMode viene aggiornato. Se, come nel nostro esempio, lo stile corrente è il nero e PutString incontra il carattere escape per lo stile corsivo, i caratteri successivi saranno visualizzati in nero corsivo. Infatti il bit corrispondente allo stile nero rimane a 1 mentre quello per lo stile corsivo passa da 0 a 1. L'unica eccezione a questa regola si verifica per il carattere escape che identifica lo stile "tondo". In questo caso tutti i bit dei diversi stili contenuti in currentMode vengono azzerati, in modo che il sistema torni allo stile tondo (quello normale, senza alterazioni della fonte carattere).

Quando PutString termina la visualizzazione della stringa, currentMode non viene alterato. La successiva chiamata a PutString, o a una qualunque delle altre routine di output su schermo, visualizzerà i caratteri ancora nello stile precedentemente impostato. È quindi una buona regola, se si desiderano stili particolari, che le stringhe in output abbiano come primo byte il carattere escape per lo stile tondo, in maniera da disabilitare qualsiasi precedente selezione, e nei byte successivi i caratteri escape che impostano gli stili desiderati.

Il carattere escape di cambio della fonte carattere

PutString può anche incontrare il carattere escape di cambio della fonte carattere all'interno di una stringa di output. Per cambio della fonte carattere s'intende anche solo la selezione di un nuovo set di caratteri (corpo) all'interno della stessa fonte. Questo carattere escape consiste nel valore 23 (decimale), seguito da una word che indica l'identificatore del set di caratteri (da qui in poi Font ID) e un byte per specificare lo stile. Il Font ID dev'essere sempre seguito da un carattere escape di controllo dello stile, dal momento che l'applicazione, nel momento in cui incontra un carattere escape, ha bisogno di sapere quali stili sono attivati. Come abbiamo già anticipato, PutString non è in grado di effettuare il cambio della fonte carattere perché dovrebbe possedere la capacità di allocare nuovo spazio di memoria, ma dal momento che i caratteri escape non sono stati creati per essere usati solo da questa routine, abbiamo ugualmente inserito un carattere escape per il cambio della fonte. PutString lo ignora, e ignora anche il byte seguente di impostazione degli stili.

I caratteri escape di cambio della posizione

Con il termine "cursore" intendiamo sempre la posizione sullo schermo su cui GEOS sta scrivendo, sia quando il cursore è visibile (input dall'utente) sia quando non lo è (output all'utente). Esistono alcuni caratteri escape che modificano la posizione corrente del cursore sullo schermo. HOME porta il cursore all'angolo superiore sinistro dello schermo (le coordinate x e y vengono azzerate). UPLINE muove il cursore verso l'alto di un numero di pixel pari all'altezza della fonte carattere adottata. GOTOX e GOTOY e GOTOXY portano il cursore su una specifica posizione. La word che segue GOTOX diventa la nuova coordinata x del cursore (0 - 319). Il byte che segue GOTOY diventa la sua nuova coordinata y (0 - 199). GOTOXY porta il cursore sulla posizione di coordinate x e y. La word che segue indica la coordinata x, e il successivo byte la coordinata y.

La tavola seguente illustra i caratteri escape disponibili e la loro sintassi.

I caratteri escape nei testi

Carattere	ASCII	Funzione
NULL	0	Carattere che conclude la stringa
BACKSPACE	8	Cancella il carattere precedente la cui larghezza e' memorizzata in lastWidth. Se l'applicazione non si preoccupa di riaggiornare lastWidth, BACKSPACE non puo' essere utilizzato una seconda volta dopo aver cancellato l'ultimo carattere. Quindi per cancellare non solo l'ultimo carattere, ma anche i precedenti, l'applicazione deve aggiornare la variabile lastWidth ogni volta che viene usato BACKSPACE. Attenzione che lastWidth non e' una variabile normalmente disponibile alle applicazioni. Il suo indirizzo esadecimale in GEOS V1.2 e V1.3 e' $8807
FORWARDSPACE	9	Muove il cursore verso destra, di un numero di pixel pari alla larghezza della fonte carattere correntemente selezionata
LF (line feed)	10	Muove il cursore verso il basso di una riga (il numero di pixel per calcolare la posizione della riga sotto quella corrente si ottiene dalla variabile curHeight, che contiene il corpo della fonte carattere selezionata)
HOME	11	Muove il cursore all'angolo sinistro in alto dello schermo

SEGUE

SEGUE

UPLINE	12	Muove il cursore verso l'alto di una riga (il numero di pixel per calcolare la posizione della riga sopra quella corrente si ottiene dalla variabile curHeight, che contiene il corpo della fonte carattere selezionata)
CR (carriage return)	13	Muove il cursore all'inizio della riga successiva: la coordinata x viene aggiornata con il valore contenuto nella variabile leftMargin, e viene impartito automaticamente il comando LF per passare alla riga successiva
ULINEON	14	Attiva la sottolineatura dei caratteri
ULINEOFF	15	Disattiva la sottolineatura dei caratteri
ESC_GRAPHICS	16	I byte che seguono costituiscono la stringa grafica che viene passata a GraphicsString
ESC_RULER	17	Indica la presenza di una riga di definizione (ruler). PutSting ignora questo carattere escape
REVON	18	Attiva la visualizzazione dei caratteri in negativo
REVOFF	19	Disattiva la visualizzazione dei caratteri in negativo
GOTOX	20	Utilizza la word che segue il comando come nuova coordinata x del cursore
GOTOY	21	Utilizza il byte che segue il comando come nuova coordinata y del cursore
GOTOXY	22	Utilizza la word che segue il comando come nuova coordinata x del cursore e il successivo byte come nuova coordinata y del cursore
NEWCARDSET	23	Indica il cambio della fonte carattere. La word che lo segue identifica il Font ID, e il byte successivo lo stile con il quale continuare a scrivere. Dal momento che PutString non e' in grado di gestire il cambio della fonte carattere, la routine ignora il comando e i tre byte successivi
BOLDON	24	Attiva la visualizzazione dei caratteri in nero
ITALICON	25	Attiva la visualizzazione dei caratteri in corsivo
OUTLINEON	26	Attiva la visualizzazione dei caratteri in outline
PLAINTEXT	27	Attiva la visualizzazione dei caratteri in tondo, disattivando tutti i precedenti stili di scrittura. I caratteri ora sono quelli della fonte carattere selezionata, senza alcuna modifica

PutString, i_PutString

Funzione: Visualizza una stringa di caratteri sullo schermo.

Indirizzo:

Putstring	$C148
i_PutString	$C1AE

Accede a:

dispBufferOn
- bit 7 se è impostato a 1 scrive sullo schermo principale
- bit 6 se è impostato a 1 scrive sul buffer di schermo

leftMargin
rightMargin
windowTop
windowBottom — se la stringa durante la visualizzazione oltrepassa uno qualunque di questi limiti, sia orizzontale sia verticale, GEOS chiama la routine puntata dal vettore stringFaultVec. Se questo vettore è azzerato, i caratteri che oltrepassano questi limiti non vengono visualizzati

Parametri:

Normali

- r0 indirizzo della stringa a terminazione nulla
- r1H coordinata y della posizione in cui inizia la visualizzazione del testo (0 - 199)
- r11 coordinata x della posizione in cui inizia la visualizzazione del testo (0 - 319)

Inline

i parametri appaiono subito dopo l'istruzione jsr

- .word coordinata x della posizione in cui inizia la visualizzazione del testo (0 - 319)
- .byte coordinata y della posizione in cui inizia la visualizzazione del testo (0 - 199)
- .byte "Testo della stringa",0

Restituisce:

- r11 coordinata x del successivo carattere da visualizzare
- r1H coordinata y del successivo carattere da visualizzare

Distrugge: a, x, y, r0L, r2 - r10, r12, r13

Sinossi: PutString visualizza nella posizione data da x e y, la stringa di caratteri puntata da r0, che dev'essere una stringa a terminazione nulla, cioè conclusa dal carattere NULL. PutString visualizza il testo utilizzando la fonte carattere correntemente selezionata. Per cambiare la fonte carattere, prima di chiamare PutString si devono utilizzare le routine LoadCharSet o UseSystemFont. PutString accetta tutti i caratteri accettati da PutChar, e inoltre anche diverse sequenze di byte escape. Ogni sequenza di escape è composta da un carattere ASCII non stampabile seguito da uno o più byte (a seconda del numero di parametri richiesti dal comando). La gestione dei testi è illustrata anche in appendice.

PutString non è in grado di interpretare il comando NEWCARDSET e i suoi parametri quando lo incontra lungo una stringa di caratteri da visualizzare. Infatti PutString è incapace di gestire la memoria disponibile per creare lo spazio in cui caricare la nuova fonte carattere.

Attenzione: PutString non effettua alcun controllo sulla correttezza dei caratteri escape presenti all'interno della stringa. I caratteri ASCII dall'8 al 27 vengono considerati caratteri escape, mentre i valori esterni a questo range e minori del valore 32 producono sicuramente il blocco del sistema.

PutDecimal

Funzione: Visualizza sullo schermo un numero binario da 16 bit in notazione decimale.

Indirizzo: $C184

Accede a: dispBufferOn
- bit 7 se è impostato a 1 scrive sullo schermo principale
- bit 6 se è impostato a 1 scrive sul buffer di schermo

leftMargin
rightMargin
windowTop
windowBottom — se il numero da visualizzare oltrepassa con una delle cifre uno qualunque di questi limiti, sia orizzontale che verticale, viene chiamata la routine puntata dal vettore stringFaultVec. Se il puntatore è azzerato i caratteri che oltrepassano questi margini non vengono visualizzati

Parametri:

a — Formato
- bit 7 impostato a 1 giustifica il numero a sinistra; impostato a 0 giustifica il numero a destra
- bit 6 impostato a 1 sopprime gli zeri rimanenti; impostato a 0 visualizza gli zeri rimanenti
- bit 0 - 5 contengono la larghezza del campo in cui deve apparire il numero quando si giustifica a destra

r0 — il numero da visualizzare, composto da 16 bit
r11 — coordinata x del punto in cui visualizzare il numero (0 - 319)
r1H — coordinata y del punto in cui visualizzare il numero (0 - 199)

Restituisce:
r11 — coordinata x del successivo carattere da visualizzare
r1H — coordinata y del successivo carattere da visualizzare

Distrugge: a, x, y, r0, r2 - r10, r12, r13

Sinossi: PutDecimal converte un numero binario in decimale e lo visualizza in caratteri ASCII utilizzando PutChar. PutChar, come sarà descritto più avanti, visualizza un carattere alla volta. Se viene selezionata la giustificazione a destra del numero, allora, attraverso i bit da 0 a 5 dell'accumulatore dev'essere passata la larghezza del campo. La posizione del pixel più a destra (il punto dal quale i caratteri iniziano a essere visualizzati verso sinistra) può essere calcolata sommando la larghezza del campo alla coordinata x passata attraverso r11.

L'input di stringhe

L'input di stringhe di caratteri dall'utente, al livello più semplice, è svolto dalla routine GetString. È interessante analizzare il flusso delle operazioni principali che questa svolge, anche per puntualizzare ancora una volta che cosa significa parlare di un sistema operativo a gestione di eventi. GetString, quando viene mandata in esecuzione, aggiorna i vettori keyVector e stringFaultVec rispettivamente con gli indirizzi della routine di elaborazione dei caratteri e della routine di gestione degli errori di visualizzazione. Il precedente indirizzo contenuto in keyVector viene provvisoriamente salvato. Oltre a queste due operazioni fondamentali, GetString visualizza, tramite una chiamata alla routine PutString, la stringa eventualmente contenuta nel buffer previsto dall'applicazione. Questa caratteristica permette all'applicazione di generare l'input guidato di una stringa, facendo apparire il contenuto del buffer nel campo di input. In questo modo l'utente ha di fronte una possibile risposta già pronta. Questa operazione viene effettuata, per esempio, quando l'utente chiede a deskTop di cambiare il nome del disco. Si apre allora una finestra, all'interno della quale appare un messaggio per l'utente, il nome corrente del disco e il cursore lampeggiante alla fine del nome.

Quando GetString ha compiuto le operazioni fondamentali, abilita il cursore visibile e restituisce il controllo a MainLoop (attraverso l'applicazione). A questo punto, ogniqualvolta MainLoop registra l'arrivo di caratteri dalla tastiera, esegue attraverso il vettore keyVector la routine impostata da GetString. Questa routine elabora il carattere passato dall'ultima esecuzione di InterruptMain. Una volta che il carattere è stato analizzato e memorizzato nel buffer, la routine restituisce il controllo a MainLoop, in modo che GEOS possa dedicare il suo tempo all'elaborazione di altri eventi (menu, pressioni del pulsante del mouse, icone...). Quando la routine puntata da keyVector si accorge che è stato premuto il tasto RETURN, inserisce nel buffer il carattere conclusivo nullo (0), disabilita il cursore visibile, ripristina keyVector, azzera stringFaultVec, e cede il controllo alla routine dell'applicazione il cui indirizzo era stato memorizzato nel vettore keyVector. È a questo punto che l'applicazione riceve la stringa in input dall'utente e può elaborarla. Quello che abbiamo appena descritto è il flusso primario delle operazioni svolte da GetString.

È importante tenere ben presente che GetString non trattiene per sé il controllo della CPU, ma si limita ad aggiornare i vettori keyVector e stringFaultVec, in modo che venga eseguita la routine di elaborazione dei caratteri (quando MainLoop riprende il controllo) solo se l'utente preme un particolare tasto. Così, se l'utente sta scrivendo il nome di un file e decide di rinunciare all'operazione prima di premere il RETURN, ha la facoltà di spostare il mouse sull'icona CANCEL e interrompere l'input del nome. Questa sinergia fra eventi diversi (input di una stringa e selezione di un'icona, per esempio) è possibile soltanto perché MainLoop continua ad avere il controllo della CPU. Vediamo come vengono combinate di solito le routine PutString e GetString.

Normalmente accade che PutString viene utilizzata per visualizzare un messaggio all'utente, e GetString riceve la risposta sotto forma di una stringa di caratteri, salvandola in un buffer temporaneo. PutString restituisce la posizione (x e y) del successivo carattere visualizzabile. Questa è la posizione alla quale verrebbe visualizzato il carattere successivo. Aggiungendo uno spazio di due pixel orizzontali (almeno) alla coordinata x restituita da PutString, si ottiene una buona posizione sullo schermo per visualizzare l'eco dei caratteri inviati in risposta dall'utente.

GetString, oltre alle coordinate x e y del punto in cui visualizzare l'eco dei caratteri immessi dall'utente, richiede altre informazioni. Per prima cosa deve sapere dove si trova il buffer temporaneo allocato dall'applicazione per memorizzare la stringa in input, e la sua lunghezza. Mentre l'utente inserisce i caratteri, GetString li memorizza nel buffer, e quando l'utente preme il tasto RETURN, termina la stringa ricevuta nel buffer con il carattere NULL (0). GetString richiede che sia specificato il massimo numero di caratteri che può ricevere dall'utente. Questo numero non è altro che la dimensione del buffer. Se l'utente digita tanti caratteri da riempire il buffer, GetString accetta solo caratteri come il Backspace e il Return, altrimenti non compie alcuna operazione e lascia inalterato il buffer.

Un altro errore che si verifica facilmente riguarda l'eco dei caratteri sullo schermo. Se l'utente inserisce tanti caratteri che l'eco prodotto oltrepassa il margine destro indicato da rightMargin, PutChar (utilizzata per la generazione dell'eco) esegue, tramite il vettore stringFaultVec, la routine di elaborazione della condizione d'errore. Questa routine può essere quella indicata dall'applicazione o può essere quella di default prevista da GEOS. L'applicazione ha quindi la facoltà di fare eseguire a GEOS una particolare routine quando l'utente batte un numero di caratteri tale da oltrepassare il margine destro indicato da rightMargin. Se la routine di elaborazione dell'errore di visualizzazione è quella messa a disposizione da GEOS, le operazioni compiute per risolvere il problema sono molto semplici. La routine non fa altro che cambiare il parametro che indica il numero massimo di caratteri che il buffer deve ricevere. Il nuovo valore corrisponde al numero di caratteri che GEOS è riuscito a visualizzare prima che si verificasse l'errore. In questo modo viene diminuita la lunghezza del buffer, e dal momento che GetString, quando il buffer è pieno, non compie alcuna operazione, non si presenta più alcun errore di visualizzazione.

GetString

Funzione: Accetta una stringa di caratteri in input dall'utente.

Indirizzo: $C1BA

Chiama: PutString, per visualizzare la stringa eventualmente presente nel buffer passato dall'utente prima di abilitare l'input

Accede a:

dispBufferOn

bit 7	se è impostato a 1 scrive sullo schermo principale
bit 6	se è impostato a 1 scrive sul buffer di schermo
bit 5	se è impostato a 1, provvisoriamente GetString scrive sia sullo schermo principale sia sul buffer di schermo, ma quando la visualizzazione dell'eco del carattere si conclude, ripristina in dispBufferOn il valore impostato dall'applicazione

leftMargin
rightMargin
windowTop
windowBottom — se la stringa da visualizzare in eco sullo schermo oltrepassa uno di questi limiti, sia orizzontale sia verticale, GEOS chiama la routine puntata dal vettore stringFaultVec. Nel caso che il valore del puntatore sia 0, i caratteri successivi che oltrepassano uno dei margini non vengono visualizzati. Un'eccezione riguarda i margini verticali windowTop e windowBottom. Se una parte del carattere in eco oltrepassa uno dei due margini verticali, rimane visibile solo la parte del carattere rimasta entro i margini. Per esempio, provate a selezionare un qualunque file con l'opzione Info da deskTop, e a scrivere un testo molto lungo. Vi accorgerete che i caratteri dell'ultima riga in basso, saranno solo parzialmente visibili, e non si sovrappongono ai lati della cornice che racchiude lo spazio per i commenti

Parametri:

keyVector	indirizzo della routine da chiamare quando l'utente ha completato l'input premendo RETURN
r0	indirizzo del buffer nel quale GetString memorizza la stringa ricevuta dall'utente

r1L	byte di flag. Se il bit 7 è impostato a 1, GEOS accede alla word r4 per puntare la routine che gestisce gli errori di visualizzazione
r1H	coordinata y in pixel del punto in cui la routine inizia a visualizzare l'eco dei caratteri (0 - 199)
r2H	massimo numero di caratteri che la routine può ricevere dall'utente
r11	coordinata x in pixel del punto in cui la routine inizia a visualizzare l'eco dei caratteri sullo schermo (0 - 319)
r4	(opzionale) indirizzo della routine che reagisce agli errori di visualizzazione

Restituisce: Niente

Distrugge: a, x, y, r0 - r13

Sinossi: GetString è facile da utilizzare e particolarmente adatta per le applicazioni che richiedono un input di caratteri da parte dell'utente. L'applicazione può passare a GetString una stringa a terminazione nulla nel buffer puntato da r0. GetString, prima di procedere all'input dei dati, visualizza il contenuto del buffer e sposta il cursore dopo l'ultimo carattere del buffer. In questo modo l'applicazione può offrire all'utente una stringa di input guida, cioè di riferimento. Se l'applicazione non desidera passare a GetString alcuna stringa guida, deve ricordare di memorizzare uno 0 come primo carattere del buffer.

GetString preleva i caratteri immessi dall'utente e li memorizza nel buffer puntato da r0. Quando l'utente preme RETURN, la stringa si conclude con il carattere NULL e viene eseguita la routine il cui indirizzo è contenuto in keyVector. Questa routine gestisce la stringa in input che GetString ha ricevuto e memorizzato nel buffer.

I caratteri inseriti dall'utente vengono visualizzati in eco sullo schermo. GetString preleva i parametri x e y contenuti in r11 e r1H, quindi visualizza l'eco dei caratteri partendo dalla posizione così individuata. GetString richiede anche che sia specificato in r2L il massimo numero di caratteri che deve accettare dall'utente. Se l'utente immette un numero eccessivo di caratteri, la routine non li memorizza nel buffer e non compie nessun'altra operazione.

Nel caso che l'eco dei caratteri oltrepassi il margine destro indicato da rightMargin, la routine PutChar, impiegata per la generazione dell'eco, non visualizza il carattere che oltrepassa il margine, ed esegue la routine indicata dal vettore stringFaultVec, senza più intervento da parte di GetString. È la routine che deve porre rimedio all'errore di visualizzazione. Il vettore stringFaultVec deve già essere stato preparato da GetString. Se l'applicazione ha impostato a 1 il bit 7 del parametro r1L, GetString memorizza nel

vettore stringFaultVec l'indirizzo della routine puntata dal parametro r4 che l'applicazione ha opportunamente aggiornato. Se invece l'applicazione ha impostato a 0 il bit 7 del parametro r1L, GetString memorizza in stringFaultVector l'indirizzo di una propria routine di gestione dell'errore. Quindi l'applicazione ha due possibilità, nel caso che si presenti un errore di visualizzazione: può prevedere una routine propria che se ne occupi, o può lasciare il compito a GEOS. Nel secondo caso, GEOS non fa altro che limitare il numero di caratteri che il buffer può contenere a quelli che sono riusciti a essere visualizzati senza oltrepassare il margine destro. Così il buffer non è più "troppo pieno", dal momento che è stato variato il numero massimo di caratteri che deve ricevere, e GetString si comporta come se avesse ricevuto il numero massimo di caratteri possibile.

6 LE ROUTINE PER LA GESTIONE DEI CARATTERI

Per molte applicazioni le routine PutString e GetString sono in grado di soddisfare pienamente tutte le esigenze legate alla gestione dei testi. Ma in realtà, queste due routine sono state ideate per l'elaborazione di un testo al livello più semplice, e non sono sufficienti per operare con i caratteri in modo più complesso. Dobbiamo quindi svincolarci dalla gestione delle stringhe, e inoltrarci nella manipolazione diretta dei caratteri, sia in input sia in output. A questo proposito GEOS prevede una serie completa di routine per aumentare la flessibilità del sistema.
Queste sono le possibilità offerte:

1) leggere e scrivere un carattere alle coordinate specificate
2) stabilire la posizione del cursore visibile (una barra verticale)
3) effettuare il cambio della fonte carattere
4) richiedere quale larghezza e quale altezza ha un particolare carattere del corpo selezionato nella fonte corrente.

Sfruttando solo queste quattro opzioni è possibile creare un sofisticato word processor. Per dare un esempio di come possono essere combinate insieme, proviamo a utilizzarle per creare una semplice versione di GetString e della relativa routine di gestione dell'evento finale. Per differenziarla da GetString possiamo chiamarla OurGetString, mentre chiameremo OurInGetString la routine che gestirà effettivamente l'evento. In questo caso "gestione dell'evento" significa leggere i caratteri provenienti dall'utente e memorizzarli nel buffer, visualizzare e aggiornare la posizione del cursore visibile, e generare l'eco dei caratteri sullo schermo. Quando la routine sarà completata nei suoi tratti fondamentali, si potrà ampliarla e renderla più sofisticata ottenendo anche che interpreti i caratteri di controllo. Un altro ritocco, inoltre, le potrebbe permettere di leggere da un buffer, anziché da tastiera, creando così un editor di testi.

Iniziamo prendendo in considerazione keyVector e keyData. Il primo contiene l'indirizzo di una routine di gestione della tastiera che GEOS esegue quando analizza il relativo input. L'altro vettore viene invece aggiornato da InterruptMain con il codice ASCII del tasto premuto dall'utente. Ogni volta che si preme un tasto, MainLoop manda in esecuzione la routine puntata da keyVector. All'inizio il vettore contiene il valore 0, in modo che qualunque tasto premuto dall'utente venga ignorato. L'applicazione che vuole analizzare i caratteri provenienti dalla tastiera, deve aggiornare keyVector con l'indirizzo di una propria routine che gestisca i caratteri ricevuti in input dall'utente. Nel nostro esempio, keyVector conterrà l'indirizzo della routine OurInGetString, e verrà aggiornato da OurGetString.

Quando l'utente preme un tasto, la routine InterruptMain che in genere gestisce gli interrupt di GEOS, aggiorna la variabile keyData con il codice del carattere premuto. InterruptMain compie questo controllo ogni sessantesimo di secondo. Durante MainLoop, GEOS accede a un flag, preventivamente elaborato da InterruptMain, per sapere se è stato premuto un tasto dall'utente, e in caso positivo chiama la routine puntata da keyVector (nel nostro caso OurInGetString). Essa può accedere a keyData per elaborare il carattere inviato dall'utente.

In effetti, InterruptMain compie anche altre operazioni. Se l'applicazione mantiene il controllo del processore per troppo tempo, non può essere eseguita MainLoop in quanto il 6510 non è in grado di gestire due operazioni contemporaneamente. Di solito i codici dell'applicazione compiono operazioni brevi, e restituiscono il controllo del processore a MainLoop dopo pochissimo tempo. Può però accadere che MainLoop non venga eseguita per un tempo abbastanza lungo, e che l'utente riesca nel frattempo a premere più di un tasto. Sembra quindi che si possano perdere alcune informazioni provenienti dalla tastiera. GEOS risolve questo problema mantenendo in un buffer (coda) tutti i caratteri che MainLoop non è riuscita ancora a leggere. Se InterruptMain riceve un carattere dalla tastiera e osserva che MainLoop non ha ancora avuto la possibilità di elaborare l'ultimo ricevuto, salva il codice del carattere in un buffer interno. In seguito, la routine di servizio della tastiera (OurInGetString) può chiamare la routine GetNextChar per svuotare il buffer interno della routine di interrupt. Ogni volta che viene eseguita, GetNextChar restituisce il successivo carattere memorizzato nel buffer di interrupt. Quando il buffer è stato interamente svuotato, GetNextChar restituisce il valore 0.

OurInGetString legge il primo carattere dalla variabile keyData, e successivamente preleva i rimanenti caratteri dal buffer di interrupt tramite un loop che chiama ciclicamente GetNextChar. Ogni volta che OurInGetString riceve un carattere, lo memorizza nel nostro buffer di input, inBuffer, e ne genera l'eco sullo schermo. Per eseguire quest'ultima operazione, OurInGetString utilizza la routine PutChar. PutChar richiede il codice del carattere da visualizzare e le coordinate x e y del punto in cui disegnarlo sullo schermo. La posizione può essere una qualunque di quelle ammesse: (0 - 319) per la coordinata x e (0 - 199) per la coordinata y. PutChar è utilizzata per questo scopo anche da GetString e PutString.

È anche possibile utilizzare stringFaultVec per manipolare i caratteri che eccedono i limiti dello schermo o i margini prestabiliti. Quando PutChar tenta di visualizzare un carattere che oltrepassa uno dei margini fissati da leftMargin e da rightMargin, chiama la routine puntata da stringFaultVec. PutChar si preoccupa anche di mascherare, durante la visualizzazione, le parti di carattere che oltrepassano i limiti verticali impostati da windowTop e windowBottom. Quindi, i caratteri che si trovano lungo i due limiti verticali possono apparire troncati. Questa caratteristica si può utilizzare, ad esempio, per effettuare lo scroll verticale di un testo all'interno di una finestra.

Si può realizzare la routine puntata da stringFaultVec (che nel nostro esempio chiameremo OurStringFault) in modo che effettui lo scroll orizzontale del testo all'interno della finestra, o in modo che mandi automaticamente a capo l'intera parola (word wrap), quando oltrepassa in parte i limiti della finestra. Nel primo caso, le variabili windowTop, windowBottom, leftMargin e rightMargin definiscono le dimensioni della finestra all'interno della quale si desidera effettuare lo scroll orizzontale di un testo di dimensioni maggiori. Quando l'utente inserisce un carattere che oltrepassa il limite destro della finestra, la routine puntata da stringFaultVec può cancellare il contenuto della finestra e visualizzare l'ultimo carattere battuto sulla stessa riga in modo da creare lo spazio per scrivere la parte destra del testo. La stessa operazione, anche se in modo più articolato e complesso, è compiuta da geoWrite per passare dal lato sinistro dello schermo a quello destro, e viceversa. Nel secondo caso quando l'utente batte un carattere che oltrepassa i limiti della finestra, la routine puntata da stringFaultVec riprende l'intera parola e la riscrive a capo, cancellandola ovviamente dalla riga precedente.

La nostra routine OurStringFault esegue quest'ultima operazione semplificandola: porta a capo solo il carattere che supera il margine, e non l'intera parola di cui fa parte. Per fare questa operazione OurStringFault deve sommare alla coordinata verticale y dell'ultima riga l'altezza della riga stessa, ottenendo così la nuova coordinata verticale y alla quale riprendere la visualizzazione dei caratteri. La posizione orizzontale è data da leftMargin. Quando la routine puntata da stringFaultVec ritorna, deve comportarsi come se fosse stata chiamata da PutChar. In effetti OurInGetString non deve accorgersi di questa chiamata. OurInGetString non si deve preoccupare se i caratteri che visualizza in eco oltrepassano i margini perché non rientra nei suoi compiti. La routine puntata da stringFaultVec deve riordinare le cose in maniera tale che OurInGetString possa continuare a svolgersi come se nulla fosse accaduto. Tutto quello che OurInGetString deve sapere è che chiamando PutChar il carattere viene visualizzato.

Per ricapitolare brevemente, OurGetString attiva il cursore visibile, azzera l'indice del buffer e aggiorna i vettori keyVector e stringFaultVec rispettivamente con gli indirizzi delle routine OurInGetSting e OurStringFault. In questo modo OurGetString inizializza l'evento. Ora, ogni volta che l'utente preme un tasto, MainLoop manda in esecuzione OurInGetString, la quale preleva i caratteri digitati accedendo a keyData e chiamando GetNextChar, memorizza i caratteri nel buffer inBuffer, produce sullo

schermo l'eco del carattere selezionato e aggiorna la posizione del cursore visibile. Se si raggiunge la fine della riga prima che l'utente abbia premuto il RETURN, OurStringFault visualizza il carattere sulla riga successiva, in modo del tutto indipendente da OurInGetString.

La routine inizia con una chiamata a PutString per visualizzare un messaggio rivolto all'utente.

```
OurGetString:
   jsr      i_PutString                    ;chiama la routine
   .word    XPOSPROMPT                     ;coordinata x (0 - 319)
   .byte    YPOSPROMPT                     ;coordinata y (0 - 199)
   .byte    "Scrivi quello che vuoi:", 0   ;messaggio da visualizzare per l'utente
   ...                                     ;il codice prosegue da questo punto
```

Adesso bisogna abilitare la visualizzazione del cursore visibile. Si devono dichiarare le dimensioni della barra verticale lampeggiante e la sua posizione iniziale sullo schermo. Dal momento che stiamo utilizzando la fonte carattere standard di GEOS, che è alta 9 punti, fissiamo in 12 punti l'altezza verticale del cursore visibile. Le coordinate x e y della posizione iniziale conviene determinarle sperimentalmente. Per adesso chiamiamo le due coordinate x e y XPOSPROMPT e YPOSPROMPT, riservandoci di valutare i valori corretti in seguito.

```
XPOSPROMPT      = valore di x (0 - 319)
YPOSPROMPT      = valore di y (0 - 199)
```

Per attivare il cursore visibile dobbiamo chiamare la routine PromptOn. Il cursore visibile è realizzato con lo sprite 1.

```
lda     #12                 ;altezza del cursore visibile
jsr     InitTextPrompt      ;inizializza il cursore visibile, ma non lo visualizza
LoadW   stringX,XPOSPROMPT  ;passa le coordinate x e y del punto in cui visualizzare
LoadB   stringY,YPOSPROMPT  ;il cursore visibile
jsr     PromptOn            ;attiva il cursore visibile
lda     #0                  ;azzera l'indice del buffer che viene utilizzato
                            ;da OurInGetString (routine che verra' descritta
                            ;nelle prossime pagine)
sta     a0                  ;a0 e' lo pseudoregistro che contiene l'indice corrente
                            ;del buffer
```

Per determinare la posizione alla quale visualizzare il cursore, PromptOn utilizza le variabili stringX e stringY. Il cursore adesso è visibile. Per finire, OurGetString deve inizializzare l'evento tramite l'aggiornamento dei vettori keyVector e stringFaultVec.

```
LoadW   keyVector, OurInGetString      ;imposta la routine di servizio
LoadW   stringFaultVec,OurStringFault  ;imposta la routine d'errore
rts                                    ;OurGetString ha impostato i vettori keyVector
                                       ;e stringFaultVec e restituisce il controllo
                                       ;a MainLoop; da questo momento, quando l'utente
                                       ;preme un carattere, MainLoop esegue
                                       ;la routine OurInGetString
```

Diamo uno sguardo più approfondito a OurInGetString. Inizia prelevando il primo carattere da keyVector e se si tratta di un ritorno carrello (RETURN) conclude la sua esecuzione azzerando i vettori keyVector e stringFaultVec, in modo che MainLoop non esegua più alcuna routine se l'utente preme altri tasti. Se il carattere non è un ritorno carrello, OurInGetString lo visualizza in eco sullo schermo tramite PutChar, e lo memorizza nel buffer di input inBuffer. Successivamente chiama GetNextChar per ricevere i caratteri in coda, finché non riceve il carattere NULL (0). A questo punto restituisce il controllo a MainLoop in modo che possano essere elaborati altri eventi, fino alla successiva pressione di un tasto da parte dell'utente. Ogni volta che OurInGetString genera l'eco del carattere sullo schermo, aggiorna anche la posizione del cursore visibile, facendolo avanzare di un numero di pixel pari alla larghezza dell'ultimo carattere visualizzato. Per ottenere questo spostamento è sufficiente aggiornare la posizione orizzontale del cursore visibile con la coordinata x indicata in r11. Dal momento che utilizziamo stringX e stringY per fissare la posizione iniziale del cursore visibile, continuiamo a utilizzarle anche per il movimento. I codici di OurInGetString sono:

```
OurInGetString:
 ldx    a0               ;preleva l'indice per puntare i caratteri nel buffer
 lda    KeyData          ;preleva il primo carattere

Charloop:
 cmp    #CR              ;controlla se l'utente ha terminato la stringa
 beq    Endstring        ;se e' cosi', termina l'esecuzione della routine
 sta    inBuffer, x      ;memorizza il carattere nel buffer di input
 pha                     ;salva il carattere
 inx                     ;punta al successivo byte del buffer di input
 MoveB  stringY, r1H     ;preleva la posizione dell'eco del carattere
 MoveW  stringX, r11     ;dalle variabili di posizione del cursore visibile
 pla                     ;recupera il valore del carattere
 jsr    PutChar          ;genera l'eco del carattere sullo schermo
                         ;PutChar restituisce la nuova posizione del carattere
                         ;successivo in r1H e r11
```

```
 MoveW   r11, stringX        ;aggiorna la coordinata x del cursore visibile
 MoveB   r1H, stringY        ;anche la coordinata y del cursore dev'essere variata,
                             ;dal momento che PutChar, tramite OurStringFault,
                             ;puo' aver mandato a capo la riga
 jsr     PromptOn            ;aggiorna la posizione del cursore visibile

 jsr     GetNextChar         ;preleva il carattere successivo dalla coda
 cmp     #0                  ;controlla se e' il carattere NULL (0)
 bne     Charloop            ;se non e' NULL ripeti

 stx     a0                  ;memorizza nuovamente l'indice del buffer per successive
                             ;chiamate di OurInGetString effettuate da MainLoop
 rts                         ;restituisce il controllo a MainLoop in modo che questa possa
                             ;dedicarsi ad altri eventi

EndString:
 lda     #0                  ;termina la stringa di input
 sta     inBuffer, x         ;contenuta in inBuffer
 LoadW   keyVector, #0       ;azzera i due vettori
 LoadW   stringFaultVec, #0
 rts
```

Questa è la struttura base per poter inserire dei caratteri da tastiera, memorizzarli in un buffer e generare l'eco sullo schermo. Può accadere che OurInGetString provi a visualizzare un carattere oltre il margine destro indicato da rightMargin. In questo caso, indipendentemente dal codice di OurGetString, PutChar cede il controllo alla routine d'errore puntata dal vettore stringFaultVec. Nel nostro esempio la routine d'errore si chiama OurStringFault. L'operazione che deve svolgere consiste nell'aggiornare le coordinate x e y del cursore visibile all'inizio della nuova riga e fare in modo che i caratteri successivi siano visualizzati a capo. Per aggiornare la coordinata y alla posizione della nuova riga, OurStringFault deve conoscere la dimensione verticale (il corpo) del set di caratteri che si sta usando. Il modo più facile per ottenere questa informazione è chiamare la routine GetRealSize. OurStringFault deve memorizzare temporaneamente il carattere che non può essere visualizzato sulla riga corrente, e ottenere la dimensione verticale chiamando GetRealSize. A questa dimensione verticale conviene aggiungere due linee di scansione per spaziare meglio le due righe. Il risultato dev'essere memorizzato in stringY. Per quanto riguarda stringX, dev'essere semplicemente aggiornata con il valore contenuto in leftMargin. A questo punto il carattere che non è stato visualizzato, perché avrebbe oltrepassato il margine destro, viene disegnato sulla nuova riga. Compiuta quest'ultima operazione, OurStringFault cede definitivamente il controllo alla routine OurInGetString, che è completamente indifferente a quanto è successo.

```
OurStringFault:
 pha                        ;salva il carattere ancora da visualizzare
 ldx    currentMode         ;lo stile corrente puo' influire sull'altezza del carattere
 jsr    GetRealSize         ;preleva la dimensione verticale del carattere
 txa                        ;l'altezza e' restituita in x
 clc
 adc    stringY             ;aggiunge l'altezza del carattere alla posizione y
                            ;della riga corrente
 adc    #2                  ;aggiunge due ulteriori linee di scansione per migliorare
                            ;la spaziatura
 sta    stringY             ;memorizza la posizione y della nuova riga
 LoadW  stringX,leftMargin  ;aggiorna la coordinata x della nuova riga
 pla                        ;riprende il carattere da visualizzare
 jsr    PutChar             ;lo visualizza all'inizio della nuova riga
 rts
```

Le due routine che abbiamo appena illustrato permettono di gestire con facilità le stringhe di caratteri in input dall'utente. La stringa in input è memorizzata nel buffer inBuffer. È relativamente semplice espandere OurGetString in modo che diventi equivalente alla funzione GetString: si tratta di fare in modo che OurGetString sia in grado di ricevere, in input dalla chiamata, l'indirizzo del buffer ove memorizzare i caratteri, il massimo numero di caratteri che il buffer può contenere, l'indirizzo della routine d'errore e quello della routine da eseguire quando finisce l'input.

Prima di procedere a illustrare nuovi argomenti, dobbiamo sottolineare una fondamentale differenza fra GetString e OurGetString. Quando l'applicazione chiama GetString, le passa anche l'indirizzo della vera e propria routine di servizio che il sistema deve eseguire quando l'utente ha premuto il RETURN. GetString memorizza l'indirizzo di questa routine nella variabile string. Se l'utente non batte il RETURN, dal punto di vista dell'applicazione non avviene alcun evento, dal momento che i suoi codici non vengono eseguiti. Solo quando viene premuto il RETURN l'evento si completa e la routine di servizio dell'applicazione può accedere al buffer per leggere l'input. Per fare in modo che OurGetString si comporti allo stesso modo, dobbiamo inserire dopo EndString e prima dell'istruzione rts la chiamata alla routine di servizio dell'evento.

Dopo aver illustrato come può avvenire la gestione delle stringhe in input dall'utente, il passo successivo consiste nell'introdurre il cambio dello stile di scrittura della fonte carattere e del set di caratteri. Prima di introdurre questi nuovi argomenti presentiamo le routine impiegate nell'esempio che abbiamo appena trattato.

GetNextChar

Funzione: Restituisce un carattere dalla coda di tastiera. Quando la coda è stata interamente svuotata, la routine restituisce il valore 0.

Indirizzo: $C2A7

Parametri: Nessuno

Restituisce: a carattere proveniente dalla coda di tastiera, o 0 se la coda è vuota

Distrugge: x

Sinossi: GEOS memorizza in un buffer i caratteri provenienti dalla tastiera. Quando il loop di MainLoop è in ritardo, per qualunque ragione, l'utente può digitare più di un carattere prima che MainLoop abbia avuto il tempo di leggere l'ultimo. Ogni volta che GetNextChar viene eseguita, restituisce in a il codice ASCII del carattere prelevato dalla coda di tastiera. Se la coda è vuota, a viene restituito con il valore 0. Normalmente le applicazioni devono essere in grado di ricevere tutti i caratteri digitati dall'utente, e quindi copiano il buffer interno di GEOS nei loro buffer opportunamente predisposti, chiamando la routine GetNextChar finché questa non restituisce il valore 0.

InitTextPrompt

Funzione: Prepara le dimensioni del cursore visibile (la barra verticale che individua il punto in cui viene visualizzato il nuovo carattere).

Indirizzo: $C1C0

Parametri: a dimensione verticale in pixel del cursore visibile

Restituisce: Niente

Distrugge: a, x, y

Sinossi: Per visualizzare il cursore visibile si usa di solito una barra verticale lampeggiante, del tipo considerato nell'analisi di GetString. InitTextPrompt crea il cursore visibile della dimensione verticale specificata in a. Il cursore viene realizzato con lo sprite 1. InitTextPrompt prepara, ma non visualizza, il cursore visibile.

PromptOn

Funzione: Visualizza il cursore visibile alle coordinate stringX e stringY.

Indirizzo: $C29B

Parametri:

stringX	coordinata x, in pixel, alla quale attivare il cursore visibile (0 - 319)
stringY	coordinata y, in pixel, alla quale attivare il cursore visibile (0 - 199)

Restituisce: Niente

Distrugge: a, x, r3L, r5L, r6

Sinossi: PromptOn attiva il cursore visibile (sprite 1) visualizzandolo alla posizione individuata da stringX e stringY. Il cursore visibile dev'essere già stato inizializzato con una chiamata alla routine InitTextPrompt.

PromptOff

Funzione: Disattiva il cursore visibile sullo schermo.

Indirizzo: $C29E

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, x, r3L

Sinossi: PromptOff serve per disabilitare il cursore visibile. Per usarla si devono adottare alcuni accorgimenti. Chiamare PromptOff con la seguente procedura:

```
php
sei
jsr      PromptOff
LoadB    alphaFlag, 0
cli
plp
```

alphaFlag contiene alcuni bit che governano il cursore visibile. Se il bit 7 è impostato a 1 il cursore visibile è attivato e il bit 6 ne sorveglia lo stato. Mentre il cursore lampeggia, il bit 6 contiene l'informazione di acceso o spento. Quando è impostato a 0 il cursore visibile è invisibile (non è un gioco di parole). I restanti 6 bit costituiscono un contatore per determinare il ritmo del lampeggio.

Chiamando PromptOff si disabilita il cursore visibile. Impostando alphaFlag a 0, GEOS non riattiva il cursore.

PutChar

Funzione: Visualizza un carattere sullo schermo. Dispone di speciali caratteri escape, per il controllo della visualizzazione.

Indirizzo: $C145

Accede a: dispBufferOn
bit 7 se è impostato a 1 scrive sullo schermo principale
bit 6 se è impostato a 1 scrive sul buffer di schermo
currentMode il carattere è visualizzato con lo stile corrente

Parametri:
a codice ASCII del carattere (0 - 96)
r11 coordinata x del punto in cui viene visualizzato il carattere (0 - 319)
r1H coordinata y del punto in cui viene visualizzato il carattere (0 - 199)

Restituisce:
r11 coordinata x della posizione in cui viene visualizzato il carattere successivo
r1H coordinata y della posizione in cui viene visualizzato il carattere successivo

Distrugge: a, x, y, r0, r2 - r10, r12 - r13

Sinossi: PutChar visualizza un carattere ASCII sullo schermo nella posizione individuata dalle coordinate x e y, ed è anche in grado di interpretare alcuni (ma non tutti) caratteri escape per il controllo della stampa sullo schermo. Essa restituisce in r11 e r1H le coordinate x e y del punto in cui sarà visualizzato il nuovo carattere. Chiamando nuovamente PutChar (digitando cioè un altro carattere), questo verrà visualizzato subito dopo il precedente.

Sono molti i caratteri escape riconosciuti da PutString, che PutChar non è in grado di interpretare. In particolare si tratta dei caratteri escape composti da due o più byte, dal momento che PutChar accetta un carattere alla volta nel registro a.

Caratteri escape riconosciuti da PutChar

Carattere	ASCII	Funzione
NULL	0	Carattere che conclude una stringa
BACKSPACE	8	Cancella il carattere precedente, la cui larghezza e' memorizzata in lastWidth ($8807). Se l'applicazione non aggiorna lastWidth con la larghezza del carattere che precede quello appena cancellato, questa funzione di controllo opera solo la prima volta
FORWARDSPACE	9	Muove il cursore verso destra di uno spazio
LF (line feed)	10	Muove il cursore di una riga verso il basso (il numero di pixel viene prelevato dalla variabile curHeight)
HOME	11	Muove il cursore all'angolo superiore sinistro dello schermo
UPLINE	12	Muove il cursore di una linea verso l'alto (il numero di pixel viene prelevato dalla variabile curHeight)
CR (carriage return)	13	Muove il cursore all'inizio della riga successiva: la coordinata x e' aggiornata con il valore contenuto in leftMargin e viene automaticamente eseguito un LF
ULINEON	14	Attiva lo stile sottolineato
ULINEOFF	15	Disattiva lo stile sottolineato
REV_ON	18	Attiva lo stile negativo
REV_OFF	19	Disattiva lo stile negativo
BOLDON	24	Attiva lo stile nero
ITALICON	25	Attiva lo stile corsivo
OUTLINEON	26	Attiva lo stile outline
PLAINTEXT	27	Disattiva ogni stile precedentemente attivato. Lo stile con il quale i caratteri sono visualizzati diventa il tondo

Se si vuole visualizzare un carattere fuori dal range imposto da leftMargin e rightMargin, PutChar chiama la routine d'errore puntata dal vettore stringFaultVec. L'applicazione deve prevedere una propria routine d'errore, altrimenti il carattere che oltrepassa i margini non viene visualizzato. Di solito la routine d'errore realizzata dall'applicazione controlla il valore contenuto in windowBottom e, se possibile, visualizza i caratteri sulla riga successiva. Ma questa è un'operazione semplice. La routine d'errore può essere anche molto più sofisticata: per esempio può riportare a capo le parole che oltrepassano i margini. PutChar effettua anche alcuni controlli con i limiti verticali windowTop e windowBottom. Se una qualunque parte del carattere da visualizzare oltrepassa i limiti windowTop o windowBottom, quella parte viene mascherata. In questo modo la visualizzazione di caratteri può essere limitata a una finestra senza che i limiti siano oltrepassati.

SmallPutChar

Funzione: Visualizza il carattere e aggiorna la coordinata x.

Indirizzo: $C202

Parametri:
a codice ASCII del carattere da visualizzare
r11 coordinata x della posizione in cui visualizzare il carattere (0 - 319)
r1H coordinata y della posizione in cui visualizzare il carattere (0 - 199)

Restituisce:
r11 coordinata x della posizione in cui visualizzare il carattere successivo
r1H invariato

Distrugge: a, x, y, r0, r2 - r10, r12, r13

Sinossi: SmallPutChar è una subroutine di PutChar. A differenza di PutChar, non prevede alcun controllo sulla posizione del carattere da visualizzare, ed è quindi l'applicazione a dover provvedere in modo autonomo. Non è in grado d'interpretare i caratteri escape di controllo e quindi si possono passare solo caratteri visualizzabili. Dopo aver visualizzato il carattere, SmallPutChar provvede ad aggiornare la coordinata x, contenuta in r11, per determinare quale dovrà essere la posizione del carattere successivo sulla stessa riga. Si tratta di una routine che manipola i caratteri con grande rapidità.

GetRealSize

Funzione: Restituisce le dimensioni, verticale e orizzontale, del carattere selezionato, tenendo conto dello stile prescelto.

Indirizzo: $C1B1

Parametri:
- a il codice ASCII del carattere ($20 - $60)
- x lo stile prescelto. Il byte deve avere la stessa struttura della variabile currentMode

Restituisce:
- y larghezza del carattere
- x altezza del carattere
- a offset dalla linea di base del carattere

Distrugge: Niente

Sinossi: GetRealSize restituisce l'altezza e la larghezza di un carattere del set correntemente selezionato, tenendo conto dello stile specificato nel registro x. L'offset dalla linea di base del carattere è la distanza dalla linea di base alla massima altezza presente nella fonte. Il calcolo per ottenere questa distanza è: BaseLineOffset = (altezza totale – discendente). GEOS mantiene questo valore nella variabile baselineOffset, e lo aggiorna a ogni cambio di corpo o fonte.

Ecco gli stili che cambiano le dimensioni dei caratteri.

Nero (Bold): aumenta la larghezza dei caratteri di un pixel

OutLine: aumenta l'altezza dei caratteri di due pixel e quindi aggiunge anche due pixel alla dimensione BaseLineOffset

Corsivo (Italic): inclina i caratteri. Deforma un ideale rettangolo contenente il carattere, facendolo diventare un parallelogramma. La larghezza non varia. Le linee di scansione che compongono il carattere vengono spostate a destra di un pixel ogni due linee, iniziando dalla linea di base. La linea di base rimane inalterata, le due linee sovrastanti vengono spostate a destra di un pixel, le due successive vengono spostate a destra di due pixel rispetto alla posizione originale e così via. Le linee al di sotto della linea di base vengono spostate verso sinistra.

GetCharWidth

Funzione: Restituisce la larghezza del carattere selezionato, tenendo conto dello stile impostato nella variabile currentMode.

Indirizzo: $C1C9

Accede a: currentMode lo stile correntemente impostato

Parametri: a il codice ASCII del carattere ($20 - $60)

Restituisce: a larghezza del carattere, oppure 0 se il carattere è un carattere di controllo (char < $20)

Distrugge: y

Sinossi: Questa routine restituisce la larghezza del carattere tenendo conto dello stile impostato.

Il controllo dello stile

Visualizzare i caratteri in nero, corsivo o uno qualunque degli altri stili è un'operazione semplice. È sufficiente passare a PutChar il relativo carattere escape per impostare lo stile desiderato. I caratteri escape sono descritti nella scheda della routine PutChar. PutChar si rende conto che si trova di fronte a un carattere escape e non prova a visualizzarlo sullo schermo. Si limita ad aggiornare la variabile currentMode per impostare il nuovo stile. Ogni bit della variabile currentMode rappresenta uno stile selezionabile. Passando a PutChar un carattere di controllo per attivare un nuovo stile, viene impostato a 1 il corrispondente bit nella variabile currentMode. Così possono essere selezionati più stili contemporaneamente. Passando il carattere di controllo corrispondente allo stile tondo (plaintext) si ottiene l'azzeramento di tutti i bit di currentMode, e quindi la disattivazione di tutti gli stili.

Qualche problema in più lo presentano le variazioni di fonte. L'applicazione dev'essere in grado di manipolare le fonti carattere in modo autonomo. Deve saper interpretare un comando di cambio della fonte inserito all'interno del testo, ma deve inoltre saper riconoscere un comando diretto dall'utente, espresso selezionando la voce di un menu o un'icona.

Una stringa di caratteri visualizzata da PutString non ha limiti nel numero di caratteri escape che può contenere. Ogni stile di scrittura, escluso il tondo, si aggiunge a quelli già selezionati. Lo stile tondo rimuove tutti quelli già impostati. Per esempio, la seguente stringa di caratteri:

```
TextExample: .byte   PLAINTEXT,"Tondo,",BOLDON," Nero,",ITALICON," Nero e
                     Corsivo.",PLAINTEXT,0
```

visualizza la seguente scritta:

Tondo, **Nero**, ***Nero e Corsivo***.

TextExample conclude la sua visualizzazione lasciando azzerati tutti gli stili, cioè selezionando lo stile tondo.

Quando si preleva una stringa in input dall'utente tramite la routine GetString, tutti i caratteri non direttamente stampabili, a parte il carattere BACKSPACE, vengono filtrati (non hanno alcun effetto e non vengono memorizzati nel buffer). Se l'applicazione sta invece ricevendo i caratteri dalla tastiera tramite la variabile keyData e la routine GetNextChar, tutti i caratteri ASCII riescono a passare, compresi quelli con il bit 7 impostato a 1, ottenuti con la pressione simultanea del tasto Commodore. Quindi se l'utente preme il carattere di controllo che attiva lo stile nero (ASCII 24 ottenuto con il tasto CTRL e il tasto x) e l'applicazione sta prelevando la stringa con la nostra routine OurGetString, l'eco sullo schermo dei caratteri successivi sarà prodotto con l'aggiunta dello stile richiesto. Di solito si vuole che OurGetString

filtri alcuni caratteri non visualizzabili, perché non vengano passati a PutChar, che li interpreterebbe nel loro significato ASCII.

Aggiungere il comando del cambio della fonte all'interno di un buffer di memorizzazione di un testo è semplice come inserire il comando di cambio di stile. È sufficiente inserire nel testo il carattere escape appropriato e i parametri che richiede. Benché esista, in ambiente GEOS, una sintassi per il comando di cambiamento delle fonti carattere, mancano le routine in grado di eseguire il comando. Per farlo, il Kernel di GEOS dovrebbe destinare un'opportuna quantità di memoria libera alle fonti carattere presenti su disco. Non sapendo a priori quali sono gli spazi disponibili e se sono sufficienti per contenere il set di caratteri selezionato (i set di caratteri possono occupare anche molta memoria), il Kernel di GEOS non può manipolare autonomamente il cambio di fonte. Nonostante questo, abbiamo ugualmente costruito una sintassi standard per il comando, in modo che particolari applicazioni non incontrino problemi di compatibilità nella gestione delle fonti carattere. Le applicazioni devono prevedere il comando di cambio della fonte nei loro buffer interni e routine in grado di interpretarlo. Approfondiremo questo punto nel prossimo paragrafo.

Le fonti carattere

In ambiente GEOS, la dimensione che caratterizza un set di caratteri è l'altezza massima misurata in punti, il cosiddetto "corpo carattere". Viene comunemente assunto che la dimensione di un punto grafico sia 1/72 di pollice. Per ragioni di comodità pratica, in ambiente GEOS un punto grafico corrisponde a un pixel dello schermo, e un pixel misura 1/80 di pollice. Si tratta della grandezza che meglio si presta a essere impiegata con le stampanti in commercio, che per la maggior parte possiedono testine in grado di stampare 80 punti per pollice.

In GEOS un set di caratteri è composto al massimo da 96 caratteri(†). In conformità con lo standard inglese, i caratteri visualizzabili sono individuati dai codici ASCII che vanno da 32 a 127. I set di caratteri opportunamente creati per lingue straniere possono contenere alcuni simboli particolari e non possederne altri. La loro larghezza è variabile (la w, per esempio, è più larga della i), ma tutti hanno necessariamente la stessa altezza massima, che identifica il "corpo" del set di caratteri. In GEOS non ci sono limiti per le possibili altezze, mentre la larghezza non può superare i 54 pixel.

Una "fonte carattere" è costituita da molti set di caratteri, diversi e di diverse dimensioni, ma accomunati dallo stesso stile dominante. La fonte di sistema, residente nel Kernel di GEOS, si chiama BSW ed è composta da un solo set di caratteri in corpo 9 (ricordiamo che il corpo non è altro che l'altezza massima misurata in pixel).

(†) Un'eccezione a questa regola: la fonte carattere di sistema BSW 9 contiene 97 caratteri, e non 96 come tutte le altre. Il codice ASCII #128 corrisponde al marchio Commodore.

Uno specifico set di caratteri all'interno di una fonte carattere è definito dal nome della fonte e dal corpo dei caratteri: BSW 9. La fonte carattere di sistema è sempre residente in memoria ed è parte integrante del Kernel di GEOS. Tutte le altre fonti disponibili sono memorizzate su disco e all'occorrenza possono essere caricate in memoria.

L'identificatore della fonte carattere

A ogni fonte carattere è associato un numero d'identificazione unico che per brevità chiamiamo ID. L'ID è un numero che occupa 10 bit. Quindi i numeri "consentiti" possono variare nel range 0 - 1023.

L'identificatore del set di caratteri

Ogni set di caratteri, nell'ambito di una fonte, viene individuato tramite un numero che ne indica il corpo, e che occupa 6 bit. La combinazione dell'ID, e del numero d'identificazione del corpo, occupa 16 bit. Questo numero rappresenta l'identificatore del set di caratteri, e individua sia il numero della fonte (ID) sia il corpo del singolo set di caratteri all'interno della fonte stessa. Chiamiamo per brevità "Font ID" l'identificatore del set di caratteri.

Font ID: bit 15/6 = ID, bit 5/0 = corpo del set

La struttura dei file delle fonti carattere

Le fonti carattere sono memorizzate in file con struttura VLIR di tipo FONT (per maggiori informazioni sulla struttura VLIR, consultare il capitolo dedicato ai file in ambiente GEOS). Ogni set di caratteri della fonte corrisponde a un record del file, secondo la struttura VLIR. Il numero che identifica il set (cioè il corpo) è anche il numero del record nel quale il set viene memorizzato. Se la fonte carattere contiene un set da 12 punti, questo viene memorizzato nel record 12 (i record sono numerati a partire dal record 0, cioè quello individuato dai primi due byte dopo l'indirizzo T/S del blocco successivo). I record ai quali non corrispondono set di caratteri nella fonte restano inutilizzati; i loro puntatori T/S contengono rispettivamente i valori $00 e $FF. Le informazioni necessarie per utilizzare la fonte carattere sono memorizzate nel blocco File Header associato al file. Lo spazio utilizzato per queste informazioni viene normalmente impiegato per contenere dati che per un file di tipo FONT non hanno significato.

Il File Header delle fonti carattere

Offset all'interno del File Header		Numero di byte	Descrizione
O_GHSET_LENGTHS	= 97 ($61)	33	Dimensioni in byte, memorizzate in word, di ogni corpo della fonte, dal piu' piccolo al piu' grande; le word sono sequenziali e il buffer dev'essere riempito con un opportuno numero di zeri
O_GHFONTID	= 130 ($82)	2	Font ID del primo corpo della fonte, il piu' piccolo
O_GHPOINT_SIZES	= 132 ($84)	28	Font ID di tutti i corpi della fonte che seguono; il buffer dev'essere riempito con un opportuno numero di zeri

Le posizioni dei byte specificate nella tavola includono l'indirizzo T/S; il primo byte del blocco (l'indirizzo di traccia) si trova in posizione 0.

Come si può osservare nella tavola appena illustrata, il numero d'identificazione della fonte carattere (ID) è memorizzato nei byte 130 e 131 del File Header. Questo numero, memorizzato sotto forma di word, è anche l'identificatore del corpo principale della fonte (Font ID). Dal byte 132 del File Header inizia una tavola di 28 byte (14 word) contenente il Font ID per ogni altro set di caratteri disponibile per la fonte. Per esempio, la fonte Elmwood, le cui caratteristiche sono riportate in appendice, contiene due corpi carattere, il 18 e il 36. L'ID della fonte è il numero 20. La combinazione dell'ID e del corpo del primo set della fonte crea il primo Font ID. In questo caso il Font ID è $0512. Questa word dev'essere memorizzata a O_GHFONTID (secondo la solita regola del byte basso e byte alto: $12, $05). A O_GHPOINT_SIZES devono essere memorizzati i Font ID dei corpi che seguono nella fonte. In questo caso abbiamo solo un secondo corpo il cui Font ID è $0524.

Riassumendo: tutti i Font ID relativi a ogni corpo della fonte, devono essere memorizzati, dal più piccolo al più grande, a partire dal byte O_GHFONTID del File Header. Queste word devono essere sequenziali e i byte che restano devono essere azzerati. I numeri di byte occupati in memoria da ogni set di caratteri sono memorizzati nella tavola da 33 byte che inizia al byte 97 del File Header. Nel nostro esempio questa tavola conterrebbe due word sequenziali, ognuna delle quali indica la lunghezza in

byte del set di caratteri corrispondente. Entrambe le tavole, la tavola dei Font ID e la tavola delle dimensioni in byte dei vari corpi, devono essere completate con zeri. Dal momento che, per ogni set, le word vengono occupate sequenzialmente a partire dalla prima, le ultime, che non corrispondono ad alcun set, devono essere azzerate. La tavola che inizia in posizione O_GHPOINT_SIZES ha una lunghezza massima di 28 byte, corrispondenti a 14 word; aggiungendo la word memorizzata a O_GHFONTID, risulta che 15 è il numero massimo di set di caratteri che una fonte può contenere.

Come si utilizzano le fonti carattere

Un set di caratteri, appartenente a una fonte, viene allocato in memoria come un blocco continuo di dati. Una volta che il set desiderato è stato caricato, per selezionarlo è sufficiente chiamare la routine LoadCharSet con il registro r0 che contiene l'indirizzo in memoria del set. Per selezionare il set di caratteri di sistema, sempre residente in memoria, si deve chiamare la routine UseSystemFont. Le applicazione possono impiegare le fonti carattere in molti modi diversi. Se l'applicazione deve usare un solo set di caratteri, per esempio, è sufficiente che lo carichi in una parte della memoria appositamente allocata e non lo cancelli più. Se invece l'applicazione deve avere a disposizione diversi set di caratteri e diverse fonti (come geoWrite o geoPaint), il programmatore dovrà prevedere alcune routine interne che carichino da disco il set desiderato. A questo proposito l'applicazione deve anche prevedere un sistema di selezione delle fonti e dei set comandato dall'utente. Come standard, geoWrite e geoPaint adottano un menu per ricevere dall'utente le informazioni riguardo alla fonte e al set desiderato.

Per essere più precisi, l'applicazione lascia nelle tavole di definizione dei menu uno spazio di 19 byte azzerati per ogni voce; questo spazio corrisponde al testo che contiene l'indicatore dello stato della fonte (selezionata o non selezionata) e il nome. Il numero massimo di voci per questo menu è otto, quindi l'applicazione non è in grado di gestire sullo stesso disco più di otto fonti contemporaneamente. I primi due byte dei 19 disponibili per ogni nome di fonte, sono impiegati per descrivere all'utente lo stato della fonte. Se la fonte è correntemente selezionata, questi due byte contengono la stringa di testo " * " come marcatore. In caso contrario, questi due byte contengono due spazi. I successivi 16 caratteri sono il nome delle fonte carattere, mentre l'ultimo carattere è quello che conclude la stringa (0).

Quando viene eseguita la routine d'inizializzazione dell'applicazione, viene anche fatta una chiamata alla routine di sistema FindFTypes (consultare il capitolo dedicato ai sistema dei file per una descrizione di questa routine), che crea una lista dei file di tipo FONT (nel nostro caso) presenti su disco. I primi otto nomi, corrispondenti alle prime otto fonti carattere presenti su disco, vengono memorizzati in uno spazio opportunamente predisposto all'interno della tavola di definizione del menu. All'apertura del menu i nomi delle fonti presenti sul disco appaiono all'interno delle otto voci.

Il passo successivo che l'applicazione deve compiere per creare il supporto di gestione delle fonti carattere, è la creazione di una tavola in memoria che contenga i corpi disponibili per ogni fonte, e la verifica che lo spazio richiesto in memoria dai set di ogni fonte sia compatibile con le esigenze dell'applicazione. L'applicazione deve prelevare queste informazioni dal blocco File Header della fonte carattere presente su disco. Ogni volta che l'utente seleziona una voce del menu di scelta delle fonti, l'applicazione ricostruisce dinamicamente il sotto-menu che elenca i corpi disponibili per quella fonte, e lo visualizza. Il tipo di sotto-menu impiegato per questa operazione è il DYN_SUB_MENU. Si realizza una struttura di menu tale che ogni sotto-menu è di tipo DYN_SUB_MENU e accede alla stessa routine di preparazione.

Vediamo come agisce la routine di gestione dinamica dei sotto-menu. Innanzi tutto deve sapere da quale delle otto voci del menu è stata selezionata, per poter identificare il nome della fonte carattere. La routine riceve dal Kernel di GEOS il numero della voce selezionata, e lo utilizza come indice all'interno della struttura del menu, per prelevare la stringa corrispondente al nome della fonte. Questo nome sarà utilizzato per individuare sul disco la fonte carattere. A questo punto la routine accede alla tavola precedentemente creata dalla routine d'inizializzazione, nella quale sono memorizzati tutti i corpi disponibili per ogni fonte. Il numero della voce selezionata viene impiegato ancora come puntatore per individuare i corpi disponibili, e creare così il sotto-menu di scelta dei corpi della fonte.

Quando l'utente ha effettuato anche la scelta del corpo, la routine di servizio associata alle voci del sotto-menu riceve il numero della voce attivata. Questo numero individua il corpo prescelto, e quest'ultima informazione non è altro che il numero del record, all'interno della struttura VLIR della fonte, che contiene il set desiderato dall'utente. Individuato il record, la routine di servizio lo carica in memoria. A questo punto il set di caratteri richiesto dall'utente può essere attivato. Per esempio, quando l'utente apre il menu di scelta delle fonti e seleziona la fonte "Roma", la routine di servizio dinamica associata a tutte le voci del menu viene a sapere quale voce è stata scelta e preleva il nome della fonte. In seguito crea il sotto-menu contenente i vari corpi disponibili per quella fonte, prima che GEOS lo visualizzi. Quando queste operazioni sono state compiute, il sotto-menu viene visualizzato e l'utente può fare la sua scelta. La routine di servizio associata a tutte le voci del sotto-menu non deve far altro che determinare quale voce è stata selezionata, e realizzare un indice che punti, all'interno della struttura del sotto-menu, al valore del corpo prescelto.

Ora l'applicazione possiede il nome della fonte e il numero del record da caricare in memoria. Deve quindi chiamare la routine OpenRecordFile passando il nome del file e il numero del record. Una volta che il set di caratteri è in memoria, come ultima operazione dev'essere chiamata la routine LoadCharSet che lo attiva e lo rende disponibile per l'utente.

In appendice appare la lista di tutte le fonti carattere attualmente disponibili.

LoadCharSet

Funzione: Attiva un particolare set di caratteri già residente in memoria.

Indirizzo: $C1CC

Parametri: r0 indirizzo della fonte carattere in memoria

Restituisce: Niente

Distrugge: a, y, r0

Sinossi: I set di caratteri sono caricabili da disco e richiedono uno spazio di memoria compreso fra 733 byte e 3970 byte. Diversi set possono risiedere in memoria simultaneamente, anche se provengono da fonti diverse. Per segnalare a GEOS quale fonte residente in memoria è correntemente attivata, viene eseguita LoadCharSet. Il registro r0 deve contenere l'indirizzo al quale è stata caricata la fonte.

UseSystemFont

Funzione: Seleziona la fonte carattere di sistema BSW 9, sempre residente in memoria.

Indirizzo: $C14B

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, y, r0

Sinossi: La fonte carattere BSW 9 è sempre residente in memoria. Dal momento che non ci sono indicatori che segnalano quale fonte è attualmente disponibile, è stata realizzata la routine UseSystemFont che offre un metodo rapido per selezionare la fonte carattere di sistema.

7 I DRIVER DI INPUT

Il driver di input standard

Attualmente GEOS dà la possibilità di utilizzare il joystick (il driver standard di input), il mouse proporzionale, la tavoletta grafica e la penna ottica. Sullo schermo, i comandi impartiti da questi dispositivi si riflettono in movimenti del mouse grafico (più semplicemente mouse). Il mouse sullo schermo è costituito da una freccia orientata verso l'alto e verso sinistra. Quando con la parola mouse si vorrà intendere il dispositivo hardware da tavolo, sarà esplicitamente dichiarato. Conveniamo quindi che la parola mouse si riferisca alla freccia presente sullo schermo. Nel corso del capitolo adotteremo la parola "dispositivo" per indicare uno qualsiasi dei dispositivi hardware esterni sopra citati.

A ogni chiamata di interrupt, la routine InterruptMain presente nel Kernel di GEOS dialoga con il driver di input, anch'esso residente in memoria. Il lavoro che il driver deve svolgere consiste nel calcolare i valori delle seguenti variabili.

```
mouseXPos word    coordinata x del mouse sul campo visibile dello schermo (0 - 319)
mouseYPos byte    coordinata y del mouse sul campo visibile dello schermo (0 - 199)
mouseData byte    bit 7  impostato a 0 se il pulsante del mouse e' premuto, impostato a 1
                         se il pulsante del mouse non e' premuto
pressFlag byte    bit 5  (MOUSE_BIT) impostato a 1 se il pulsante del mouse cambia di stato
                  bit 6  (INPUT_BIT) impostato a 1 se dall'ultima chiamata di interrupt e'
                         variata la posizione del mouse hardware
```

Sia le applicazioni sia il Kernel di GEOS possono leggere i valori contenuti in queste variabili e agire di conseguenza. Il Kernel accede al bit 5 (MOUSE_BIT) della variabile pressFlag per determinare se lo stato del pulsante ha subito una variazione. Se la variazione c'è stata, il Kernel accede alla variabile mouseData per sapere se l'evento

consiste in una pressione o in un rilascio del pulsante. Se il pulsante del mouse è stato premuto (mouseData diventa positivo), GEOS controlla se il mouse si trova sopra un'icona o un menu, e in questo caso esegue la routine di servizio associata. Altrimenti esegue la routine puntata dal vettore otherPressVec.

Ma GEOS, cosa abbastanza curiosa, gestisce anche il rilascio del pulsante del mouse. Infatti, quando si accorge che il suo stato è passato da premuto a rilasciato (mouseData diventa negativo), esegue ancora la routine puntata dal vettore otherPressVec. È importante sottolineare che questa gestione avviene anche quando il mouse si trova su una regione significativa dello schermo. La routine puntata da otherPressVec viene *sempre* eseguita: sia quando il mouse si trova su un'icona, o vi si trovava al momento della precedente pressione, sia quando si trova su una zona non significativa. Quindi, la routine puntata da otherPressVec dev'essere in grado di distinguere le pressioni del pulsante del mouse dai rilasci, e riconoscere se un rilascio si verifica dopo una pressione su una zona significativa oppure no.

Questa caratteristica, che può apparire curiosa, si deve in realtà al fatto che la gestione del rilascio del pulsante del mouse è stata aggiunta relativamente in ritardo al Kernel di GEOS. I moduli di gestione dei menu e delle icone erano già completi. Però questo apparente bug offre in effetti un vantaggio: l'applicazione può reagire al rilascio anche quando la precedente pressione è avvenuta in una zona significativa dello schermo. In genere questa esigenza non si presenta, e quindi le routine puntate da otherPressVec ignorano il rilascio del pulsante del mouse in qualunque posizione avvenga, o lo tengono presente solo quando avviene con il mouse non sovrapposto a un menu o a un'icona. Per fare un esempio, geoWrite impiega la gestione del rilascio per definire un blocco di testo quando l'utente mantiene premuto il pulsante del mouse.

I compiti del driver di input

Il driver di input ha principalmente il compito di "leggere" lo stato dell'hardware attraverso i byte hardware presenti nel C-64, e riportarne il valore nelle variabili mouseData e pressFlag. Il driver deve registrare il cambiamento di posizione del mouse e memorizzare le nuove coordinate nelle variabili mouseXPos e mouseYPos.

Dal momento che i dispositivi hardware di input sono molto diversi fra loro, anche i metodi di accesso dei vari driver di input e i parametri significativi cambiano sostanzialmente caso per caso. Per fare un esempio, il driver per il joystick accede all'hardware leggendo la porta del joystick, e in seguito calcola l'accelerazione nella direzione indicata dal joystick. Dall'accelerazione può calcolare velocità e posizione finale. Nel caso di un mouse proporzionale, la gestione dei movimenti è totalmente diversa. Il mouse Commodore 1351 invia in output alla porta del joystick livelli di tensione diversi, e il processore SID, presente all'interno del C-64, legge il livello di tensione memorizzando un numero da 8 bit per entrambe le coordinate x e y. Il driver

di input di gestione del mouse proporzionale legge questi due byte e ricava la nuova posizione del mouse. Non occorrono altri esempi per riconoscere quanto diversi possono essere i driver di input per i vari tipi di dispositivi hardware. Queste differenze devono comunque essere trasparenti per le applicazioni, alle quali interessa esclusivamente che a ogni generazione di interrupt siano aggiornate le quattro variabili principali precedentemente citate. L'hardware di input adottato e il relativo tipo di driver di input, sono informazioni che devono risultare completamente invisibili per l'applicazione. Riducendo in questo modo i parametri di colloquio fra applicazione GEOS e dispositivo hardware di input, si raggiunge la massima compatibilità del sistema GEOS e delle applicazioni con qualsiasi tipo di dispositivo hardware di input.

Locazioni fondamentali dei driver di input

Lo spazio di memoria che GEOS mette a disposizione per contenere i codici del driver di input selezionato è di 380 byte e inizia all'indirizzo MOUSE_BASE ($FE80 - $FFF9). Appena installato, il sistema operativo GEOS contiene il driver di input standard del joystick. Quando l'utente seleziona un nuovo dispositivo hardware di input, all'indirizzo MOUSE_BASE viene caricato il corrispondente driver, disallocando completamente il driver precedente. Se si desidera realizzare un driver di input personalizzato, si deve rilocarlo all'indirizzo individuato da MOUSE_BASE.

Tutte le applicazioni GEOS compatibili si aspettano che il driver di input correntemente in memoria contenga tre routine fondamentali: InitMouse, SlowMouse e UpdateMouse. Oltre a queste routine fondamentali, le applicazioni si aspettano che il driver di input mantenga aggiornate le quattro variabili fondamentali: mouseXPos, mouseYPos, mouseData e pressFlag. Le tre routine fondamentali devono svolgere i loro compiti prescindendo dal tipo di driver di input al quale appartengono, e possono essere comunque allocate nello spazio di memoria occupato dal driver di input, dal momento che questo inizia con una tavola di salto (o jump table) che le individua univocamente:

```
Indirizzo       Contenuto
MOUSE_BASE      jmp   InitMouse
MOUSE_BASE + 3  jmp   SlowMouse
MOUSE_BASE + 6  jmp   UpdateMouse
```

Queste tre locazioni sono gli indirizzi ai quali il Kernel di GEOS e le applicazioni fanno riferimento per comunicare con il driver di input. Per esempio, se il Kernel desidera chiamare la routine UpdateMouse, deve eseguire l'istruzione jsr MOUSE_BASE + 6 durante InterruptMain. La prima routine fondamentale che il driver di input deve possedere è InitMouse. Questa routine viene eseguita per inizializzare tutte le variabili interne ed esterne che il driver utilizza, prima che siano chiamate le altre routine.

InitMouse

Funzione: Inizializza la configurazione hardware gestita dal driver di input correntemente in memoria.

Indirizzo: $FE80 (MOUSE_BASE)

Parametri: Nessuno

Restituisce:

mouseXPos	impostato al valore iniziale (0 - 319)
mouseYPos	impostato al valore iniziale (0 - 199)
mouseData	bit 7 impostato a 1 per indicare che il pulsante del mouse è in stato di rilascio

Distrugge: a, x, y, r0 - r15 (conviene assumere che tutti i registri siano stati distrutti dal momento che non è possibile sapere quali variabili potrebbero aver subito dei cambiamenti con il driver di input precedente)
a, x, y, r0 - r2, r7 - r8, per il driver del joystick

Sinossi: Questa routine inizializza tutte le variabili richieste dal driver di input. Alcune variabili non sono direttamente accessibili e sono da considerare ad uso interno del driver.

Dopo aver chiamato la routine InitMouse, è possibile eseguire le routine SlowMouse e UpdateMouse.

Accelerazione, velocità e variabili non standard

Alcuni dispositivi di input, come il joystick, devono essere regolati secondo diversi livelli di sensibilità. Per esempio, qualche volta l'utente può desiderare che il mouse raggiunga la sua velocità massima in pochissimo tempo, mentre altre volte, come all'apertura di un menu, preferirebbe disporre di un mouse che si muove molto lentamente per poter agevolmente scegliere la voce desiderata, senza correre il rischio di chiudere il menu oltrepassandone i bordi.

Altri dispositivi di input, come il mouse proporzionale e la tavoletta grafica, non sono gestiti attraverso il computo della velocità e dell'accelerazione, ma interagiscono direttamente con la posizione associata del mouse sullo schermo. Altri non ancora inventati potrebbero essere gestiti con variabili del tutto diverse. Tenendo presente queste considerazioni, appare evidente la necessità di svincolare il dialogo tra applicazione e dispositivo di input dalle attuali caratteristiche delle configurazioni hardware esterne. Il problema si risolve in tre parti.

Per prima cosa, qualsiasi driver di input deve garantire alcune funzioni di base. Come minimo, queste funzioni devono mantenere aggiornate le quattro variabili fondamentali mouseXPos, mouseYPos, mouseData e pressFlag.

In secondo luogo, possono essere allocate nella RAM dedicata alle variabili del sistema operativo GEOS alcune variabili addizionali per dispositivi simili al joystick. Il joystick è il dispositivo di default per quanto riguarda l'input dall'utente, e il suo driver ha bisogno di conoscere istante per istante velocità e accelerazione del mouse. Le variabili associate a questi due parametri di spostamento includono maxMouseSpeed, minMouseSpeed e mouseAccel, e vengono aggiornate con i loro valori di default dalla routine d'inizializzazione del driver di input. Il desk accessory Preference Manager può così facilmente accedere a tali variabili per intervenire sulla velocità di spostamento e l'accelerazione del mouse. La routine SlowMouse viene eseguita dal Kernel per riportare la velocità di spostamento del mouse a zero, per esempio durante l'apertura di un menu. SlowMouse e le variabili appena citate permettono di ottenere un controllo del joystick di elevata qualità. Può sembrare un tipo di controllo troppo sofisticato per un dispositivo come il joystick, ma ricordando che viene impiegato dalla maggior parte degli utenti, riteniamo che i nostri sforzi siano più che giustificati.

Alcuni dispositivi di input, come il mouse proporzionale Commodore 1351, non richiedono speciali trattamenti. Il mouse proporzionale non si basa sulla velocità per impostare la nuova posizione, ma sulla distanza. Il suo movimento è di una tale precisione che consente di regolare molto accuratamente la posizione del mouse sullo schermo. Nel caso che in futuro diventi disponibile qualche nuovo dispositivo che necessiti di un trattamento speciale, abbiamo adottato un terzo tipo di approccio.

Se le quattro variabili fondamentali non sono sufficienti per riportare tutte le informazioni di gestione del nuovo dispositivo hardware di input, si rende necessario allocare un nuovo spazio in memoria, per accogliere eventuali nuovi parametri. GEOS

consente ai driver di input d'impiegare, a questo scopo, i quattro byte che iniziano all'indirizzo inputData. Un'applicazione che oltre alla posizione, allo stato del pulsante, alla velocità e all'accelerazione del mouse deve ricevere altre informazioni dal driver di input, le può leggere in questi quattro byte. **Nota importante:** le applicazioni che devono accedere alle variabili allocate a inputData per completare le informazioni ricevute dal driver di input, diventano dipendenti dal particolare dispositivo di input, a meno di particolari accorgimenti.

A ogni cambiamento di stato del dispositivo di input, il driver corrispondente deve svolgere le seguenti operazioni:

1) aggiornare le quattro variabili fondamentali
2) aggiornare i quattro byte a inputData, se impiegati
3) impostare a 1 il bit INPUT_BIT (bit 6) della variabile pressFlag.

Dal momento che di solito GEOS non accede a inputData, l'applicazione deve prevedere al suo interno una routine di lettura di queste variabili opzionali. L'indirizzo di questa routine dev'essere memorizzato nel vettore inputVector. Quando il Kernel di GEOS trova il bit INPUT_BIT della variabile pressFlag impostato a 1, chiama la routine puntata dal vettore inputVector (se non è azzerato). Per esempio, il driver del joystick memorizza nel primo byte e nel secondo byte a inputData la direzione e la velocità del mouse. Questi sono parametri opzionali per la gestione del mouse. geoPaint utilizza questi valori al suo interno per effettuare lo scroll del disegno. Quando l'utente seleziona il modo scrollMode, geoPaint memorizza nel vettore inputVector l'indirizzo della routine per realizzare lo scroll. Ogni volta che la direzione impostata dal joystick cambia, GEOS chiama il vettore inputVector e la routine interna di geoPaint interrompe lo scroll o ne cambia la direzione.

Questo particolare impiego opzionale delle variabili a inputData non è così semplice per tutti i dispositivi di input. Per esempio, sebbene la direzione e la velocità di spostamento del mouse siano parametri naturali per il joystick, sono completamente atipici per dispositivi come il mouse proporzionale. I driver per questi dispositivi devono generare gli appropriati parametri di direzione autonomamente, in modo che sia ancora possibile la piena compatibilità con l'applicazione geoPaint.

L'unica ragione per impiegare le variabili opzionali a inputData è la gestione di un nuovo dispositivo di input nato per comunicare con un'applicazione specifica. Dal momento che questa particolare configurazione è incompatibile con altre applicazioni e altri dispositivi di input dall'utente, la scelta di questo terzo approccio dovrebbe essere ben ponderata.

Le applicazioni possono leggere il contenuto della stringa allocata a inputDevName per conoscere il nome del driver di input in memoria in quel momento. L'interfaccia utente deskTop memorizza a inputDevName la stringa a terminazione nulla (17 byte) corrispondente al nome del driver di input.

A questo punto dovrebbe essere abbastanza chiaro quali sono i compiti fondamentali di un driver di input:

1) determinare la posizione del mouse e le variabili di gestione del pulsante
2) eventualmente aggiornare le variabili allocate a inputData
3) memorizzare a inputData i due parametri direzionali se si desidera la piena compatibilità con geoPaint.

Le variabili di cui abbiamo parlato in questa introduzione saranno discusse in dettaglio dopo la presentazione delle routine SlowMouse e UpdateMouse.

La routine SlowMouse

Come abbiamo già sottolineato, la routine SlowMouse imposta a zero la velocità del mouse, che in seguito può di nuovo aumentare. Non tragga in inganno il nome: la funzione di questa routine non è affatto quella di ridurre la velocità massima, indicata dalla variabile maxMouseSpeed.

SlowMouse esiste principalmente per rendere più facile all'utente l'apertura di un menu o la selezione tra le sue varie voci. Quando l'utente apre un menu e seleziona una voce, il Kernel di GEOS apre il sotto-menu corrispondente e posiziona il mouse sulla prima voce. L'utente può poi selezionare una di queste voci. Ci siamo resi conto che la maggior parte degli utenti tengono il joystick inclinato nella direzione in cui deve apparire il sotto-menu, fino a quando questo non è stato interamente visualizzato. Dal momento che GEOS, durante la visualizzazione del sotto-menu, continua ad aggiornare le variabili del mouse tramite InterruptMain, è probabile che in quel lasso di tempo il mouse raggiunga la massima velocità di spostamento; può quindi accadere che quando GEOS posiziona il mouse sulla prima voce del sotto-menu, questo ne oltrepassi i bordi chiudendolo senza che l'utente abbia avuto il tempo di leggerlo. Si è quindi reso necessario provvedere a una routine che azzeri la velocità di spostamento del mouse in modo che questa situazione non possa verificarsi. I driver per il mouse o per la tavoletta grafica non utilizzano la velocità, ma devono ugualmente far ricorso a SlowMouse, anche se in questo caso viene eseguita solo l'istruzione rts.

Se invece si desidera semplicemente ridurre la velocità di spostamento del mouse, l'applicazione deve diminuire il valore di maxMouseSpeed e mouseAccel. I valori standard per queste variabili sono riportati nelle pagine successive.

SlowMouse

Funzione: Riporta il mouse alla massima sensibilità, riducendone la velocità a zero.

Indirizzo: $FE83 (MOUSE_BASE + 3)

Parametri: Nessuno

Restituisce: Niente

Distrugge:

a	per il driver del joystick
a, x, y, r0 - r15	da assumere per qualsiasi altro driver di input

Sinossi: Quando il mouse si muove sopra un menu, diventa qualche volta necessario fermarlo azzerandone la velocità. Questa routine non riduce la massima velocità impostata dalla variabile maxMouseSpeed, ma azzera la velocità di spostamento del mouse (se il driver è per il joystick). Subito dopo il mouse può riaccelerare fino a raggiungere di nuovo la massima velocità.

Le versioni di SlowMouse attualmente disponibili nei driver della tavoletta grafica e del mouse proporzionale eseguono semplicemente l'istruzione rts.

La routine UpdateMouse

UpdateMouse è la routine principale di un driver di input. I suoi compiti includono la "lettura" della porta che realizza il collegamento hardware con il dispositivo di input per determinare eventuali cambiamenti di stato, e il successivo calcolo delle variabili mouseXPos, mouseYPos, mouseData e pressFlag. Se il driver di input vuol essere completamente compatibile con l'applicazione geoPaint, deve aggiornare anche i parametri opzionali a inputData. A questo indirizzo si ricorre anche nel caso che siano richiesti parametri di comunicazione particolari.

UpdateMouse

Funzione: Aggiorna le variabili di controllo del mouse. Viene eseguita dalla routine InterruptMain a ogni chiamata di interrupt.

Indirizzo: $FE86 (MOUSE_BASE + 6)

Parametri:

mouseXPos	valore corrente della coordinata x del mouse
mouseYPos	valore corrente della coordinata y del mouse
cia1prb	porta del joystick

Restituisce:

mouseXPos	nuova coordinata x del mouse
mouseYPos	nuova coordinata y del mouse
mouseData	bit 7 impostato a 0 per indicare che il pulsante è premuto, a 1 per indicare che il pulsante è rilasciato
pressFlag	MOUSE_BIT impostato a 1 se il pulsante del mouse ha appena cambiato il suo stato stato INPUT_BIT impostato a 1 se dall'ultima chiamata di interrupt si è verificato un cambiamento nell'input dall'utente
inputData	quattro byte aggiuntivi per parametri opzionali di gestione del driver di input. Nel caso del driver per il joystick, i primi due byte contengono i parametri direzionali
inputData:	bit 0 - 7 direzione del joystick: 0= destra 1= avanti e destra 2= avanti 3= avanti e sinistra 4= sinistra 5= indietro e sinistra 6= indietro 7= indietro e destra −1= joystick centrato
inputData + 1	velocità corrente del mouse

Distrugge: a, x, y, r0 - r15 per il joystick

Sinossi: Questa routine viene eseguita a ogni chiamata di interrupt per aggiornare la posizione del mouse sullo schermo e leggere lo stato del pulsante; mouseXPos e mouseYPos vengono aggiornate a ogni chiamata di interrupt. Quando il pulsante del mouse cambia il suo stato, il bit MOUSE_BIT in pressFlag viene impostato a 1; mouseData diventa positivo se il pulsante è premuto (bit 7 = 0), negativo se è rilasciato (bit 7 = 1). Se il bit MOUSE_BIT è stato impostato dalla routine UpdateMouse, il Kernel di GEOS legge mouseData durante il suo ciclo in MainLoop per determinare se il cambiamento di stato del pulsante caratterizza una pressione o un rilascio. GEOS in seguito chiama l'appropriata routine di servizio (che può essere per un'icona, per un menu o, attraverso otherPressVec, per la pressione del pulsante in una zona dello schermo non convenzionale) in base alla posizione del mouse sullo schermo. Al ritorno da questa routine, il bit MOUSE_BIT in pressFlag viene impostato a 0.

Informazioni speciali provenienti dal dispositivo di input o generate dallo stesso driver devono essere passate a GEOS e alle applicazioni attraverso i quattro byte disponibili all'indirizzo inputData. Il bit INPUT_BIT in pressFlag dev'essere impostato a 1 quando uno di questi parametri speciali a inputData cambia il proprio valore. Durante MainLoop, GEOS chiama la routine puntata dal vettore inputVector se il bit INPUT_BIT è impostato a 1.

Le variabili di gestione del mouse per il driver di input

Il modulo di controllo del mouse gestisce diverse variabili. Alcune di queste variabili sono già state brevemente presentate.

```
Variabili del mouse richieste

mouseXPos       word  coordinata x della posizione del mouse sullo schermo visibile
                      (0 - 319)
mouseYPos       byte  coordinata y della posizione del mouse sullo schermo visibile
                      (0 - 199)
mouseData       byte  bit 7 impostato a 1 se il pulsante e' rilasciato, impostato a 0 se
                      il pulsante e' premuto
pressFlag       byte  bit 5 (MOUSE_BIT) impostato a 1 dal driver di input se avviene
                      un cambiamento nello stato del pulsante del mouse
                      bit 6 (INPUT_BIT) impostato a 1 se dall'ultima chiamata di interrupt
                      e' avvenuto un cambiamento di stato del dispositivo di input

MOUSE_BIT       =  %00100000
INPUT_BIT       =  %01000000

Variabili  opzionali  del  mouse

maxMouseSpeed   byte  Questa variabile viene utilizzata per controllare la massima
                      velocita' di spostamento del mouse sullo schermo quando
                      il dispositivo di input e' il joystick. Non viene impiegata
                      dai driver per la tavoletta grafica e per il mouse proporzionale.
                      Il suo valore ottimale dipende da come il driver di input
                      determina velocita' e posizione correnti. Per il joystick
                      il range e' (0 - 127). Il valore di default e':

                           MAX_VELOCITY = 127

                      Questa e' la costante per la velocita' massima di default
                      da memorizzare in maxMouseSpeed

minMouseSpeed   byte  Questa variabile e' utilizzata per controllare la minima
                      velocita' di spostamento del mouse sullo schermo, quando
                      il dispositivo di input e' il joystick. Non viene impiegata
                      dai driver per la tavoletta grafica e per il mouse proporzionale.
                      Il suo valore ottimale dipende da come il driver di input determina
```

velocita' e posizione correnti. Per il joystick il range e' (0 - 127). Il valore di default e':

MIN_VELOCITY = 30

Questa costante viene memorizzata nella variabile minMouseSpeed. Una velocita' minima minore di questo valore puo' venire impostata per regolazioni molto precise della posizione del mouse sullo schermo, ma e' sconsigliabile per le operazioni normali

mouseAccel byte Questo byte controlla la rapidita' con la quale il mouse raggiunge la velocita' massima consentita, quando il dispositivo di input e' il joystick. Nel caso di una tavoletta grafica, questa variabile potrebbe essere utilizzata come rapporto di conversione fra lo spostamento della penna sulla tavoletta e il corrispondente spostamento sullo schermo. Attualmente questa variabile viene impiegata solo dal driver del joystick. Per il joystick il range e' (0 - 255). Il valore di default e':

MOUSE_ACCEL= 127

inputVector word Questo vettore contiene l'indirizzo della routine dell'applicazione, eseguita da MainLoop, che gestisce i parametri speciali nel caso di dispositivi di input particolari o a comunicazione diretta. L'esigenza di uno spazio supplementare per i parametri di gestione del particolare dispositivo di input, nasce dalla possibilita' che alcuni driver generino un numero maggiore di informazioni, oltre alle coordinate x e y del mouse, e che alcune applicazioni richiedano questi parametri addizionali. Se la routine UpdateMouse di un particolare driver e' piu' elaborata e permette la gestione di informazioni opzionali, deve memorizzarle nell'array da quattro byte a inputData, e deve impostare a 1 il bit INPUT_BIT nella variabile pressFlag. Quando il ciclo MainLoop di GEOS si accorge che questo bit e' impostato a 1, chiama la routine il cui indirizzo e' memorizzato in inputVector

inputData 4 byte Questa piccola area di memoria puo' essere impiegata per memorizzare parametri addizionali per la gestione di un particolare dispositivo di input. Per il joystick:
inputData: bit 0 - 7 direzioni del joystick: 0= destra
1= avanti e destra

```
 2= avanti
 3= avanti e sinistra
 4= sinistra
 5= indietro e sinistra
 6= indietro
 7= indietro e destra
-1= joystick centrato
```

```
inputData + 1: velocita' del mouse corrente
```

Il mouse visto dall'applicazione

Finora abbiamo discusso l'argomento dal punto di vista di un programmatore che vuole realizzare un driver di input. L'altra faccia della medaglia riguarda l'interazione fra l'applicazione e il driver di input. La gestione del mouse effettuata da GEOS è già stata discussa: ogni volta che l'utente preme il pulsante del mouse, GEOS controlla se il mouse si trova su un'icona, su un menu o su un'area dello schermo non convenzionale, e chiama le appropriate routine di gestione dell'evento.

Per ottenere questo tipo di gestione del mouse, si deve eseguire la routine StartMouseMode. Dal momento che è deskTop a chiamare questa routine, le applicazioni non dovrebbero aver bisogno di eseguirla. Per disattivare le funzioni del mouse si deve chiamare la routine ClearMouseMode. L'effetto è che un bit della variabile mouseOn viene azzerato, lo sprite per il mouse viene disattivato (i dati dello sprite non sono più direttamente accessibili per essere visualizzati; questo stato è molto importante per l'interfaccia RS-232, per gli accessi al disco e per tutte le routine che hanno tempi precisi da rispettare) e UpdateMouse non viene più eseguita durante i cicli di interrupt. Questa è la ragione per cui il mouse appare e scompare durante gli accessi al disco: ClearMouseMode viene eseguita dal codice di gestione del turbo. Per ripristinare il funzionamento del mouse dopo aver chiamato ClearMouseMode, si deve eseguire la routine StartMouseMode.

Per disattivare temporaneamente il disegno del mouse sullo schermo, lasciando però che le sue variabili di gestione continuino a venire aggiornate, si deve chiamare la routine MouseOff. La routine UpdateMouse nel driver di input continua a essere eseguita, e solo lo sprite 0 è temporaneamente disattivato. Per riabilitare la visualizzazione del mouse sullo schermo è sufficiente eseguire la routine MouseUp, la quale riabilita la visualizzazione del mouse alla successiva chiamata di interrupt. Durante la temporanea cancellazione del mouse dallo schermo, l'utente può continuare a muoverlo anche se non riesce a vederne gli spostamenti. Chiamando MouseUp il mouse riappare nella nuova posizione.

StartMouseMode

Funzione: Abilita il funzionamento del mouse. Questa routine chiama anche InitMouse.

Indirizzo: $C14E

Parametri:
- C — se è impostato a 1, la routine aggiorna la posizione del mouse con le coordinate ricevute, e azzera la velocità
- r11 — nuova coordinata x, abilitata solo se il carry è impostato a 1 (0 - 319)
- y — nuova coordinata y, abilitata solo se il carry è impostato a 1 (0 - 199)

Restituisce: Niente

Distrugge: a, x, y, r0 - r15

Sinossi: StartMouseMode attiva il funzionamento del mouse. Viene abilitato il vettore mouseVector. Le variabili globali mouseLeft, mouseRight, mouseTop, e mouseBottom devono essere già state predisposte. StartMouseMode viene eseguita di norma dalla routine d'inizializzazione dell'applicazione.

Se il carry è impostato a 1, le coordinate della posizione desiderata del mouse, passate attraverso r11 e y, vengono memorizzate nelle variabili mouseXPos e mouseYPos, e la routine SlowMouse viene eseguita per ridurre a zero la velocità del mouse.

ClearMouseMode

Funzione: Questa routine è complementare a StartMouseMode. Ha la funzione di disattivare completamente la gestione del mouse e quindi la sua visualizzazione sullo schermo.

Indirizzo: $C19C

Parametri: Nessuno

Restituisce: mouseOn azzerata

Sinossi: ClearMouseMode disabilita completamente la gestione del mouse da parte del Kernel di GEOS. Lo sprite sullo schermo viene cancellato tramite la routine DisableSprite, e la variabile mouseOn viene azzerata. Per riattivare il controllo e la visualizzazione del mouse si deve mandare in esecuzione la routine StartMouseMode.

MouseOff

Funzione: Disabilita temporaneamente la visualizzazione del mouse sullo schermo.

Indirizzo: $C18D

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, x, y, r3L

Sinossi: Disattiva la visualizzazione del mouse sullo schermo. Tutte le variabili di gestione del mouse continuano a essere aggiornate. Si può cancellare provvisoriamente il mouse dallo schermo chiamando MouseOff, e successivamente farlo riapparire chiamando MouseUp.

MouseUp

Funzione: Visualizza il mouse dopo una chiamata a MouseOff.

Indirizzo: $C18A

Parametri: Nessuno

Restituisce: Niente

Distrugge: a

Sinossi: Riabilita la visualizzazione del mouse dopo che è stata chiamata la routine MouseOff e imposta a 1 un flag per indicare a GEOS che alla prossima chiamata di interrupt deve ridisegnare anche il cursore visibile (se prima era abilitato). Si può cancellare temporaneamente il mouse dallo schermo chiamando MouseOff, e successivamente farlo riapparire chiamando MouseUp.

Controlli addizionali del mouse

GEOS permette alle applicazioni di limitare i movimenti del mouse a una particolare zona dello schermo. Il Kernel di GEOS costringe il mouse a non uscire dal rettangolo definito da due variabili word (mouseLeft e mouseRight) e da due variabili byte (mouseTop e mouseBottom). Il driver di input non ha bisogno di conoscere i valori di queste variabili, dal momento che si limita ad aggiornare la posizione del mouse, memorizzando le nuove coordinate nelle variabili mouseXPos e mouseYPos. Quando il Kernel di GEOS legge i valori di queste due variabili, prima di portare realmente il mouse nella nuova posizione controlla che non corrisponda a un punto esterno alla regione delimitata, e se è necessario sposta il mouse sul bordo del rettangolo. In questo caso il Kernel di GEOS chiama la routine puntata dal vettore mouseFaultVec. Inizialmente questo vettore è impostato a 0 dal Kernel. L'applicazione, per esempio, può memorizzare nel vettore mouseFaultVec l'indirizzo di una routine interna in grado di effettuare lo scroll di un documento all'interno di una finestra. Per rilevare quale spostamento del mouse ha causato l'impiego del vettore mouseFaultVec, l'applicazione può accedere alla variabile faultData, i cui possibili valori sono riportati in appendice.

Nel Kernel di GEOS è prevista anche una routine in grado di controllare se il mouse si trova all'interno di una particolare regione dello schermo. Questa routine è utile soprattutto se l'applicazione deve attribuire un significato particolare alla pressione del pulsante all'interno di una regione predefinita dello schermo, e questa regione non è controllata direttamente dal Kernel. Si tratta della routine IsMseInRegion, e i parametri che richiede sono le coordinate dei lati della regione da controllare.

C'è infine una coppia di variabili destinate alla gestione del mouse: mousePicData (contiene 63 byte che definiscono il disegno del mouse sullo schermo) e mouseVector (contiene l'indirizzo di una routine che MainLoop chiama a ogni ciclo per gestire il mouse). Se il bit MOUSEON_BIT della variabile mouseOn è impostato a 1, ogni volta che il driver di input segnala la pressione del pulsante, viene eseguita la routine puntata dal vettore mouseVector. È decisamente improbabile che il programmatore ritenga necessario intervenire sul vettore mouseVector cambiandone il contenuto, perché così disabiliterebbe anche la gestione di icone e menu da parte di GEOS. Di solito, per aggiungere funzioni alla gestione del mouse i programmatori intervengono sul vettore otherPressVec.

La variabile mouseOn contiene anche alcuni bit per attivare e disattivare la gestione dei menu e delle icone. Sfortunatamente, quando GEOS chiama la routine di gestione dei menu, questa imposta a 1 anche il bit che attiva la gestione delle icone, senza controllare se l'applicazione le aveva precedentemente definite. Per questo motivo è conveniente che ogni applicazione definisca almeno un'icona fittizia, per quanto non ne abbia effettivo bisogno.

Queste variabili e le costanti per impostarle sono descritte dopo la presentazione della routine IsMseInRegion.

IsMseInRegion

Funzione: Controlla se il mouse si trova all'interno di una particolare regione dello schermo.

Indirizzo: $C2B3

Parametri:

r2L	coordinata y del lato superiore della regione (0 - 199)
r2H	coordinata y del lato inferiore della regione (0 - 199)
r3	coordinata x del lato sinistro della regione (0 - 319)
r4	coordinata x del lato destro della regione (0 - 319)

Restituisce: a TRUE (–1) se il mouse è nella regione, FALSE (0) se è esterno alla regione

Distrugge: Niente

Sinossi: IsMseInRegion controlla se le coordinate x e y del mouse rappresentano un punto dello schermo interno alla regione specificata. Se questo accade la routine restituisce lo stato TRUE in a, altrimenti lo stato FALSE.

Le variabili di gestione del mouse per le applicazioni

Le seguenti variabili sono impiegate dal modulo di gestione del mouse, presente nel Kernel di GEOS, e sono disponibili per le applicazioni.

```
mouseOn     byte        Questa variabile contiene alcuni flag che determinano
                        lo stato del mouse. Inoltre contiene alcuni bit impiegati
                        nella gestione dei menu e delle icone
            bit 7       Se e' impostato a 1, il mouse e' abilitato
            bit 6       Se la gestione dei menu e' attivata, e' impostato a 1
            bit 5       Se la gestione delle icone e' attivata, e' impostato a 1

SET_MSE_ON  = %10000000 bit impostato a 1 in mouseOn per abilitare il mouse
SET_MENUON  = %01000000 bit impostato a 1 in mouseOn per abilitare i menu
SET_ICONSON = %00100000 bit impostato a 1 in mouseOn per abilitare le icone (un bug
                        nel Kernel imposta a 1 questo bit quando si abilitano i menu)

mouseLeft      word     il mouse non puo' spostarsi piu' a sinistra di questa posizione
                        (0 - 319)
mouseRight     word     il mouse non puo' spostarsi piu' a destra di questa posizione
                        (0 - 319)
mouseTop       byte     il mouse non puo' spostarsi piu' in alto di questa posizione
                        (0 - 199)
mouseBottom    byte     il mouse non puo' spostarsi piu' in basso di questa posizione
                        (0 - 199)

mousePicData   63 byte  Dati per il disegno dello sprite che visualizza il mouse.
                        Il Kernel di GEOS ne esegue una copia nell'area che contiene
                        i dati della figura corrente del mouse

mouseVector    word     Vettore che punta la routine chiamata dal Kernel di GEOS quando
                        il pulsante del mouse viene premuto

mouseFaultVec  word     Vettore che punta la routine chiamata dal Kernel di GEOS
                        quando il mouse tenta di uscire dalla regione specificata.
                        L'eventuale routine associata puo' accedere alla variabile
                        faultData per conoscere la causa che ha prodotto la propria
                        esecuzione. Se il vettore e' azzerato, GEOS non permette
                        al mouse di uscire dai confini stabiliti. Consultare
                        il capitolo 21 per maggiori informazioni sul vettore
                        mouseFaultVec
```

Joystick

Questo file contiene il driver di input del joystick.

Routine eseguibili: o_InitMouse
o_SlowMouse
o_UpdateMouse

```
.include geosMacros
.include geosConstants
.include geosMemoryMap
.include geosRoutines


 .psect $FD84   ;il File Header e' assemblato a $FD84 se viene impiegato il programma
                ;Basic PRGTOGEOS per trasformare un programma PRG in uno in formato GEOS
```

Blocco File Header del driver del joystick

```
JoyHdr:
          ;I primi quattro byte del File Header sono impostati dal programma
          ;di conversione PRGTOGEOS

.byte     (63|$80)                     ;64 byte per definire il disegno dell'icona
.byte     %11111111,%11111111,%11111111
.byte     %10000000,%00000000,%00000001
.byte     %11000000,%11001111,%11110001
.byte     %10100011,%00110000,%00001001
```

```
.byte     %10011100,%00000001,%10000101
.byte     %10000000,%01100010,%01000101
.byte     %10000000,%10011100,%00100101
.byte     %10000001,%01101100,%00100101
.byte     %10000010,%11100100,%00011001
.byte     %10000100,%11011010,%00000001
.byte     %10001000,%00111010,%00000001
.byte     %10010000,%00110110,%00000001
.byte     %10100000,%00001100,%11111001
.byte     %10100000,%00011000,%10000101
.byte     %10110000,%00110000,%10000101
.byte     %10011000,%01100000,%11111001
.byte     %10001100,%11000000,%10000001
.byte     %10000111,%10000000,%10000001
.byte     %10000011,%00000000,%10000001
.byte     %10000000,%00000000,%00000001
.byte     %11111111,%11111111,%11111111
.byte     $80|USER                        ;tipo Commodore associato ai file GEOS
.byte     INPUT_DEVICE                    ;tipo GEOS associato al file
.byte     SEQUENTIAL                      ;file a struttura sequenziale
.word     MOUSE_BASE                      ;indirizzo d'inizio del file
.word     EndJoystick                     ;indirizzo di fine del file (-1)
.word     0                               ;non usato attualmente (indirizzo inizio
                                          ;esecuzione)
.byte     "Input Drvr  V1.1",0,0,0,0      ;16 byte nome del file + 4 zeri
.byte     "Dave & Mike",0,0,0,0,0,0,0,0,0 ;16 byte nome autore + 4 zeri
.byte     0,0,0,0,0,0,0,0,0,0             ;16 byte nome applicazione parente + 4 zeri
.byte     0,0,0,0,0,0,0,0,0,0
.byte     0,0,0,0,0,0,0,0,0,0             ;23 byte liberi per le applicazioni
.byte     0,0,0,0,0,0,0,0,0,0
.byte     0,0,0
.byte     0                               ;inizio del testo opzionale associato
                                          ;al file, zero per nessun testo
.block    95                              ;spazio disponibile per il testo opzionale
```

Tavola di salto alle routine del driver per il mouse

```
.psect          MOUSE_BASE ;$FE80
;
;               Tavola di salto del driver di input
;
InitMouse:      jmp    o_InitMouse    ;entry #0
SlowMouse:      jmp    o_SlowMouse    ;entry #1
UpdateMouse:    jmp    o_UpdateMouse  ;entry #2
;
;               Variabili globali:
;
;               Queste variabili sono allocate nell'area a inputData prevista
;               per contenere variabili opzionali di gestione del dispositivo di input

mouseDirection  =      inputData      ;direzione corrente del mouse
mouseSpeed      =      inputData + 1  ;velocita' corrente del mouse
;
;               Variabili locali:
;
fracXMouse:     .byte  0  ;frazione della posizione del mouse
fracYMouse:     .byte  0  ;frazione della posizione del mouse
fracSpeedMouse: .byte  0  ;frazione della corrente velocita' del mouse
velXMouse:      .byte  0  ;componente x di currentSpeed
velYMouse:      .byte  0  ;componente y di currentSpeed
currentMouse:   .byte  0  ;stato corrente del pulsante del mouse
curJoyStatus:   .byte  0  ;stato corrente del joystick
lastKeyRead:    .byte  0  ;valore in input dalla porta del joystick
```

InitMouse

Chiamata da: Kernel di GEOS o applicazione, durante ogni fase d'inizializzazione

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, x, y, r0 - r15

Sinossi: Questa routine inizializza il mouse e lo posiziona nell'angolo superiore sinistro dello schermo.

```
o_InitMouse:
        jsr     o_SlowMouse         ;esegue lda #0, sta mouseSpeed
        sta     fracSpeedMouse
        sta     mouseXPos
        sta     mouseXPos + 1
        sta     mouseYPos
        lda     #-1                 ;nessuna direzione
        sta     mouseDirection
        jmp     ComputeMouseVels    ;memorizza la corrente velocita'
```

SlowMouse

Chiamata da: Kernel di GEOS quando apre un menu

Parametri: Nessuno

Restituisce: a azzerato

Distrugge: x, y, r1 - r13

Sinossi: Questa routine azzera la velocità del mouse.

```
o_SlowMouse:     LoadB  mouseSpeed, 0  ;azzera la velocita' del mouse
SM_rts:          rts
```

UpdateMouse

Chiamata da: I codici di interrupt

Parametri: Nessuno

Restituisce: mouseXPos e mouseYPos aggiornati

Distrugge: a, x, y, r0 - r15

Sinossi: Questa routine viene eseguita a ogni chiamata di interrupt per aggiornare la posizione del mouse sullo schermo. Legge la porta del joystick e registra i nuovi valori della velocità. Sulla base di queste informazioni, la routine calcola le coordinate della nuova posizione del mouse. Non muove direttamente il mouse, ma ne aggiorna le coordinate per individuarne la nuova posizione. È compito del Kernel di GEOS stabilire se visualizzare il mouse nella nuova posizione.

```
o_UpdateMouse:  jsr   C64Joystick          ;legge la porta del joystick e ne aggiorna
                                           ;la velocita'
                bbrf  MOUSEON_BIT, mouseOn, SM_rts  ;questa macro istruzione cede
                                                    ;il controllo a SM_rts se il mouse
                                                    ;e' disabilitato, senza compiere
                                                    ;alcun aggiornamento
                jsr   UpdateMouseVels      ;registra la velocita' del mouse calcolando
                                           ;anche l'accelerazione
                jsr   UpdateMouseX         ;registra la coordinata x del mouse e prosegue,
                                           ;al ritorno dalla subroutine, in UpdateMouseY
```

UpdateMouseY

Chiamata da: o_UpdateMouse

Parametri: mouseYPos coordinata y corrente del mouse

Restituisce: mouseYPos coordinata y aggiornata

Distrugge: a, x, y, r1H

Sinossi: Aggiorna la posizione y del mouse valutandola per mezzo della velocità corrente.

```
UpdateMouseY:
            ldy    #0            ;assume che la velocita' sia positiva
            lda    velYMouse     ;legge la velocita' verticale corrente
            bpl    10$           ;se positiva y = 0
            dey                  ;se negativa y = -1 ($FF)
10$:
            sty    r1H           ;memorizza il byte alto
            asl    a             ;slitta a sinistra tre volte
            rol    r1H
            asl    a
            rol    r1H
            asl    a
            rol    r1H
            add    fracYMouse    ;somma l'incremento della coordinata
            sta    fracYMouse    ;memorizza il nuovo incremento
            lda    r1H           ;preleva il byte alto della velocita'
            adc    mouseYPos     ;somma la precedente coordinata
            sta    mouseYPos
20$:
            rts
```

UpdateMouseVels

Chiamata da: o_UpdateMouse

Parametri: mouseSpeed velocità corrente del mouse
velXMouse, velYMouse componenti della velocità corrente

Restituisce: mouseSpeed, velXMouse, velYMouse aggiornate

Distrugge: a, x, y, r0 - r2

Sinossi: Aggiorna la velocità del mouse tenendo conto dell'accelerazione.

```
UpdateMouseVels:
            ldx     mouseDirection      ;legge la direzione del mouse
            bmi     20$                 ;se il joystick e' centrato effettua il branch
            lda     maxMouseSpeed       ;confronta con la massima velocita'
            cmp     mouseSpeed
            blt     15$                 ;se e' massima non cambiarla
            lda     mouseAccel          ;addiziona alla velocita' l'incremento
            add     fracSpeedMouse      ;di velocita'
            sta     fracSpeedMouse
            bcc     30$
            inc     mouseSpeed          ;se necessario incrementa la velocita' del mouse
            bra     30$
15$:
            sta     mouseSpeed
20$:
            lda     minMouseSpeed       ;preleva la velocita', che
            cmp     mouseSpeed          ;non puo' diventare meno del minimo
            bge     25$                 ;se e' minore del minimo, allora "branch" (salta)
            lda     fracSpeedMouse      ;sottrae dalla velocita' l'incremento
            sub     mouseAccel
            sta     fracSpeedMouse
            bcs     30$
            dec     mouseSpeed          ;se necessario diminuisce la velocita' del mouse
            bra     30$
```

```
25$:
            sta    mouseSpeed
30$:
            jmp    ComputeMouseVels    ;adesso i parametri sono sufficienti
                                       ;per determinare la nuova velocita'
                                       ;verticale e orizzontale
```

ComputeMouseVels

Funzione: Routine interna: calcola la velocità del mouse tramite la direzione del joystick.

Chiamata da: Viene chiamata internamente

Parametri: mouseDirection direzione del joystick
mouseSpeed velocità corrente del mouse

Restituisce: velXMouse, velYMouse aggiornati secondo la direzione assunta dal joystick

Distrugge: a, x, y, r0 - r2

Sinossi: Questa routine calcola la velocità verticale e la velocità orizzontale del mouse, sulla base della direzione assunta dal joystick.

```
ComputeMouseVels:
                ldx    mouseDirection
                bmi    10$                  ;se il joystick e' rilasciato, azzera
                                            ;le velocita'
                MoveB  mouseSpeed, r0L      ;passa i parametri
                jsr    SineCosine
                MoveB  r1H, velXMouse
                MoveB  r2H, velYMouse
                rts
10$:
                lda    #0                   ;se il joystick e' rilasciato
                sta    velXMouse            ;azzera la velocita' orizzontale
                sta    velYMouse            ;azzera la velocita' verticale
                rts
```

UpdateMouseX

Funzione: Routine interna: aggiorna la coordinata x del mouse basandosi sulla sua velocità.

Chiamata da: o_UpdateMouse

Parametri: mouseXPos coordinata x del mouse

Restituisce: mouseXPos aggiornata

Distrugge: a, x, y, r11, r12L

Funzione: Questa routine aggiorna la coordinata x del mouse basandosi sulla sua velocità orizzontale.

```
UpdateMouseX:
            ldy   #$FF       ;assume un valore negativo
            lda   velXMouse
            bmi   10$        ;se negativo salta
            iny              ;altrimenti il segno diventa positivo
10$:
            sty   r11H
            sty   r12L
            asl   a          ;moltiplica per 8 per ottenere la velocita'
                             ;permanente elevata alla terza
            rol   r11H
            asl   a
            rol   r11H
            asl   a
            rol   r11H
                             ;somma la velocita' allo spostamento
            add   fracXMouse ;somma lo spostamento
            sta   fracXMouse ;memorizza lo spostamento
            lda   r11H       ;preleva il byte basso della velocita'
```

```
adc     mouseXPos          ;somma il byte basso della velocita'
sta     mouseXPos          ;e memorizza
lda     r12L               ;si utilizza matematica in tripla precisione
adc     mouseXPos+1        ;somma il byte alto della coordinata x intera
sta     mouseXPos+1        ;r11 contiene la nuova coordinata x
rts
```

C-64Joystick

Funzione: Routine interna: legge la porta del joystick per accedere alle variabili di posizione e di movimento del mouse.

Chiamata da: o_UpdateMouse

Parametri: Nessuno

Restituisce: Niente

Altera:

lastKeyRead	contiene l'ultima lettura del joystick
curJoyStatus	impostato per contenere la nuova direzione del joystick
pressFlag	MOUSE_BIT impostato a 1 se il pulsante è premuto JOY_BIT impostato a 1 se la direzione del joystick è variata
mouseDirection	nuova direzione del joystick,
mouseData	nuovo stato del pulsante del mouse, se ci sono state variazioni

Distrugge: a, x, y

```
C64Joystick:
        LoadB  cia1pra, %11111111   ;si accerta di leggere valori dal joystick
        lda    cia1prb              ;preleva il dato del joystick A (porta 1)
        eor    #$FF                 ;dato complementare per logica positiva
        cmp    lastKeyRead          ;dev'essere lo stesso due volte
        sta    lastKeyRead          ;memorizza il valore per il confronto
        bne    20$                  ;se non e' uguale, esce

        and    #$0F                 ;isola i bit del joystick
        cmp    curJoyStatus         ;confronta con la posizione corrente
        beq    10$                  ;se non c'e' nessuna differenza, branch
        sta    curJoyStatus         ;memorizza il nuovo valore
        tay
```

```
            lda   directionTable, y   ;preleva dalla tavola il valore da passare
            sta   mouseDirection
            smbf  INPUT_BIT, pressFlag ;segnala il cambiamento avvenuto
                                      ;nel dispositivo di input
            jsr   ComputeMouseVels
10$:
            lda   lastKeyRead         ;preleva il dato per la pressione
            and   #%00010000          ;isola il bit del pulsante
            cmp   currentMouse        ;e lo confronta con il valore corrente
            beq   20$                 ;se non c'e' nessuna differenza, branch
            sta   currentMouse        ;altrimenti imposta il nuovo stato del pulsante
            asl   a                   ;e lo trasporta nel bit 7
            asl   a
            asl   a
            eor   #$10000000          ;valore di complemento per passare
                                      ;in logica positiva
            sta   mouseData
            smbf  MOUSE_BIT, pressFlag ;segnala il cambiamento intervenuto
                                      ;nello stato del pulsante
20$:
            rts


directionTable:
            .byte  -1        ;se non e' selezionata alcuna direzione, passa a=-1
            .byte  2         ;vedere la conversione hardware descritta all'inizio
            .byte  6         ;di questo modulo per comprendere
            .byte  $FF       ;la conversione della direzione qui effettuata
            .byte  4         ;notare che $FF corrisponde a uno stato non valido,
            .byte  3         ;questo stato non dovrebbe verificarsi mai, a meno
            .byte  5         ;che il joystick non sia rotto
            .byte  $FF
            .byte  0
            .byte  1
            .byte  7
            .byte  $FF
            .byte  $FF
            .byte  $FF
            .byte  $FF
            .byte  $FF
```

SineCosine

Funzione: Routine interna: SineCosine genera seno e coseno di 16 possibili direzioni e moltiplica questi valori per un parametro.

Chiamata da: ComputMouseVels

Parametri:

x, mouseDirection	direzione (0 a 15)
r0L	parametro della velocità

Restituisce:

r1H	velocità x
r2H	velocità y

Distrugge: a, x, y, r0, r6 - r8

```
SineCosine:
        lda   cosineTable, x      ;salva il valore del coseno
        sta   r1L
        lda   sineTable, x        ;salva il valore del seno
        sta   r2L
        lda   sineCosineTable, x ;preleva il segno
        pha
        ldx   #r1L                ;calcola la velocita' orizzontale
        ldy   #r0L
        jsr   BBMult
        ldx   #r2L                ;calcola la velocita' verticale
        jsr   BBMult              ;y punta gia' a r0L
        pla
        pha
        bpl   10$                 ;se la velocita' orizzontale e' positiva, branch
        NegateW   r1
10$:
        pla
        and   #%1000000
        beq   20$                 ;se la velocita' verticale e' positiva, branch
```

```
                NegateW   r2
20$:
                rts


cosineTable:
                .byte 255               ;direzione 0 - 0 gradi dell'angolo
                .byte 181               ;direzione 2 - 45 gradi dell'angolo
                                        ;nota: la tavola dei coseni si sovrappone
                                        ;a quella dei seni che segue
sineTable:
                .byte 0                 ;direzione 0 - 0 gradi dell'angolo
                .byte 181               ;direzione 2 - 45 gradi dell'angolo
                .byte 255               ;direzione 4 - 90 gradi dell'angolo
                .byte 181               ;direzione 6 - 135 gradi dell'angolo
                .byte 0                 ;direzione 8 - 180 gradi dell'angolo
                .byte 181               ;direzione 10 - -135 gradi dell'angolo
                .byte 255               ;direzione 12 - -90 gradi dell'angolo
                .byte 181               ;direzione 14 - -45 gradi dell'angolo

sineCosineTable:
                .byte POSITIVE | (POSITIVE  1)   ;direzione 0 - 0 gradi dell'angolo
                .byte POSITIVE | (NEGATIVE  1)   ;direzione 2 - 45 gradi dell'angolo
                .byte POSITIVE | (NEGATIVE  1)   ;direzione 4 - 90 gradi dell'angolo
                .byte NEGATIVE | (NEGATIVE  1)   ;direzione 6 - 135 gradi dell'angolo
                .byte NEGATIVE | (POSITIVE  1)   ;direzione 8 - 180 gradi dell'angolo
                .byte NEGATIVE | (POSITIVE  1)   ;direzione 10 - -135 gradi dell'angolo
                .byte POSITIVE | (POSITIVE  1)   ;direzione 12 - -90 gradi dell'angolo
                .byte POSITIVE | (POSITIVE  1)   ;direzione 14 - -45 gradi dell'angolo
```

8 LA GESTIONE DEGLI SPRITE

Il Kernel di GEOS prevede una semplice interfaccia per la gestione degli sprite hardware del C-64. Le routine di cui è composta controllano gli sprite scrivendo negli appositi registri presenti nel processore VIC e negli spazi di memoria in cui il VIC legge i disegni degli sprite. Il lettore dovrebbe avere familiarità con la gestione di base degli sprite messa a disposizione dal C-64. Una delle scelte fatte durante la creazione di GEOS (per quanto riguarda gli sprite), è stata d'installare esclusivamente le funzioni di gestione fondamentali. Applicazioni che richiedono un uso elaborato degli sprite, come i giochi, non hanno un particolare bisogno di GEOS, mentre le applicazioni che riguardano problemi commerciali o che trattano principalmente i testi troveranno utilissime le caratteristiche messe a disposizione da GEOS, e probabilmente non avranno bisogno di tecniche di gestione degli sprite troppo sofisticate.

Il Kernel di GEOS prevede quattro routine per disegnare, cancellare e posizionare gli sprite:

DrawSprite
PosSprite
EnablSprite
DisablSprite

DrawSprite

Funzione: Disegna/ridisegna uno sprite.

Indirizzo: $C1C6

Parametri:
r3L numero dello sprite (0 - 7)
r4 puntatore ai dati che compongono il disegno dello sprite

Restituisce: 64 byte copiati dall'area puntata da r4 all'area che GEOS dedica allo sprite il cui numero è specificato da r3L (viene copiato un byte extra, sebbene uno sprite sia composto da 63 byte)

Distrugge: a, y, r5

Sinossi: DrawSprite trasferisce i dati grafici dello sprite dall'area puntata da r4 all'area RAM che il processore VIC utilizza per visualizzare lo sprite il cui numero è passato in r3L. Lo sprite 0 viene utilizzato per il mouse, e lo sprite 1 per il cursore visibile (la barra verticale). Questa routine provvede solo a sostituire la figura corrente dello sprite con un'altra, e quindi agisce anche se lo sprite non è abilitato.

PosSprite

Funzione: Posiziona uno sprite sullo schermo utilizzando le coordinate GEOS x e y (non le coordinate hardware del C-64).

Indirizzo: $C1CF

Parametri:

r3L	numero dello sprite (0 - 7)
r4	coordinata x (0 - 319)
r5L	coordinata y (0 - 199)

Restituisce: r3L inalterato

Distrugge: a, x, y, r6

Sinossi: PosSprite converte le coordinate x e y che indicano la posizione dello sprite – in ambiente GEOS sono una word (0 - 319) per x e un byte (0 - 199) per y – nelle strane coordinate hardware tipiche del C-64. Le coordinate convertite sono memorizzate nei registri hardware di posizione presenti nel VIC, i cui valori determinano la posizione dello sprite sullo schermo.

EnablSprite

Funzione: Attiva la visualizzazione di uno sprite sullo schermo.

Indirizzo: $C1D2

Parametri: r3L numero dello sprite (0 - 7)

Restituisce: Gli appositi registri del VIC aggiornati per visualizzare lo sprite. r3L non viene alterato

Distrugge: a, x

Sinossi: Aggiorna l'opportuno bit del registro mobenble del VIC per abilitare lo sprite il cui numero viene passato in r3L.

DisablSprite

Funzione: Disabilita la visualizzazione di uno sprite.

Indirizzo: $C1D5

Parametri: r3L numero dello sprite (0 - 7)

Restituisce: Gli appositi registri del VIC vengono aggiornati per disabilitare la visualizzazione dello sprite. r3L non viene alterato

Distrugge: a, x

Sinossi: Aggiorna i bit del registro mobenble del VIC, per disabilitare lo sprite il cui numero viene passato in r3L.

9 I PROCESSI TEMPORIZZATI

GEOS è in grado di gestire i processi temporizzati. Un processo temporizzato, per il Kernel di GEOS, è una subroutine che dev'essere eseguita ogni volta che è stato effettuato un predefinito numero di chiamate di interrupt. InterruptMain imposta un flag quando, per un certo processo, il numero di interrupt incorsi corrisponde a quello assegnato al processo dall'applicazione. Il flag indica a MainLoop che deve eseguire il processo temporizzato associato. Il sistema può eseguirne molti simultaneamente (o meglio durante lo stesso ciclo di MainLoop). Per esempio, una parte dello schermo può essere destinata a un orologio, mentre un'altra è occupata dal testo che sta battendo l'utente e in una finestra un foglio elettronico ricalcola i valori di tutte le celle. I processi temporizzati si impiegano anche per costruire subroutine che reagiscono a una certa situazione operando le opportune scelte. Per esempio si potrebbe usarne uno per cambiare la figura del mouse da cursore visibile a freccia quando si muove da un testo su un'icona.

Per la gestione dei processi temporizzati sono disponibili in GEOS:

1) un modulo timer in InterruptMain per decrementare i contatori associati ai vari processi temporizzati
2) un modulo di attivazione in MainLoop per eseguire i processi temporizzati quando i rispettivi timer lo segnalano
3) l'applicazione deve provvedere a una tavola di definizione dei processi temporizzati, all'interno della quale vengono memorizzati gli indirizzi delle routine di servizio associate a ogni processo definito.

Quando il modulo di gestione dei processi viene inizializzato, a ogni processo è assegnato un timer e un flag (un byte). A ogni sessantesimo di secondo, il modulo timer presente in InterruptMain decrementa tutti i timer associati ai vari processi. Quando uno dei timer si azzera, InterruptMain imposta a 1 il bit di run presente nel flag di processo associato. Quando MainLoop, nel suo controllo periodico, trova il bit

di run di uno dei flag di processo abilitati impostato a 1, esegue la routine di servizio associata al processo. L'applicazione deve aver previsto una tavola di dati nella quale siano definiti gli indirizzi delle routine di servizio associate a ogni processo temporizzato, e il tempo che deve trascorrere fra un'esecuzione e la successiva. Ecco una tipica tavola di definizione dei processi:

```
ProcessTable:
  .word    ProcessRoutine1   ;indirizzo della routine di servizio del processo 1
  .word    N1                ;esegui ProcessRoutine1 ogni N1 chiamate di interrupt
  .word    ProcessRoutine2   ;indirizzo della routine di servizio del processo 2
  .word    N2                ;esegui ProcessRoutine2 ogni N2 chiamate di interrupt
```

Queste informazioni sono le sole di cui GEOS ha bisogno per gestire i processi temporizzati.

I processi possono trovarsi in tre stati diversi: eseguibile, bloccato, congelato. Un processo eseguibile ha il timer associato che viene continuamente decrementato a ogni chiamata di interrupt, e quando questo si azzera viene normalmente eseguita la routine di servizio. Un processo bloccato ha il timer che continua a essere decrementato a ogni chiamata di interrupt, ma quando arriva a zero la routine di servizio del processo non viene eseguita. Un processo congelato ha il conteggio del suo timer momentaneamente interrotto. Anche in questo caso, la routine di servizio associata al processo non può essere eseguita.

Vediamo come sfruttare le opportunità offerte da ciascuno dei tre stati in cui un processo può trovarsi. Il "congelamento" è molto utile quando due o più processi devono essere eseguiti in un certo ordine. Dovendo, per un motivo qualsiasi, interrompere lo svolgimento dei processi, si scoprirà che dopo aver disabilitato gli interrupt, congelato i processi e infine riabilitato gli interrupt, le relazioni fra i vari timer associati sono rimaste inalterate.

Facciamo un altro esempio. Supponiamo di predisporre un processo che visualizza un orologio ogni secondo, ma che deve interrompere la visualizzazione in particolari periodi di tempo. Bloccando il processo, il relativo timer continua a essere decrementato, ma la routine associata non viene eseguita e quindi l'orologio non viene visualizzato per tutto il periodo desiderato. Purtroppo non viene eseguito nemmeno il cambiamento dell'ora, dunque l'orologio sarà in ritardo.

Le applicazioni hanno anche la possibilità di intervenire sulla normale gestione dei processi. La routine EnableProcess manda forzatamente in esecuzione, al successivo ciclo di MainLoop, un particolare processo. Se questo è bloccato, o congelato, dopo l'esecuzione forzata rimarrà ancora bloccato o congelato. Con questa routine un processo può essere mandato in esecuzione in qualunque momento, e se è in stato di eseguibile, il suo timer inizia nuovamente a decrementarsi.

Oltre alla gestione dei processi temporizzati, GEOS consente anche di "addormentare" una qualsiasi routine. Il concetto è molto semplice. Quando una routine usufruisce di questa opportunità, interrompe momentaneamente la sua esecuzione, e nel periodo

d'interruzione possono essere eseguiti altri codici. Quando l'intervallo prestabilito è terminato, la routine riprende l'esecuzione dal punto in cui era stata interrotta. Si noti che durante l'intervallo i contenuti dei registri possono essere variati.

È sconsigliabile adottare una routine che utilizzi l'opportunità di "addormentarsi" e contemporaneamente abbia a che fare, attivamente o passivamente, con lo stato della macchina o con le variabili globali, in quanto è difficile prevedere quale sarà la nuova configurazione al momento del "risveglio". Un'altra utile precauzione è quella di fare eseguire direttamente da MainLoop le routine che utilizzano l'opzione sleep, senza intermediazioni di altre routine di livello superiore. Vediamone il motivo. Quando una routine viene risvegliata, riprende l'esecuzione nel punto esatto in cui era stata interrotta, ma viene a trovarsi in un nuovo ambiente. Quando era stata originariamente addormentata, il modulo di gestione sleep aveva memorizzato l'indirizzo dell'istruzione successiva alla jsr Sleep. Allo scadere dell'intervallo di tempo fissato, il modulo di gestione sleep esegue l'istruzione jsr all'indirizzo precedentemente salvato. La routine riprende quindi l'esecuzione (si risveglia) e si completa secondo il suo naturale decorso. Ma a questo punto, eseguendo l'istruzione rts che la termina, la routine non restituisce il controllo alla routine di livello superiore che l'aveva inizialmente chiamata, ma al modulo di gestione sleep presente in MainLoop. In generale solo le routine chiamate da MainLoop dovrebbero utilizzare l'opzione sleep, altrimenti c'è il rischio che riprendano la loro esecuzione in un ambiente totalmente diverso. Per concludere, è bene tener presente che anche avere una routine addormentata e nel frattempo addormentarne un'altra è sconsigliabile. Potrebbero crearsi dei cicli completamente indesiderati e difficili da analizzare. L'opzione sleep è utile per visualizzare messaggi sullo schermo che non alterano lo stato della macchina. Per esempio, supponiamo che una routine debba visualizzare diversi messaggi sullo schermo, facendo in modo che ogni messaggio rimanga visualizzato per un certo tempo e l'utente riesca a leggerlo. Normalmente questa operazione si può realizzare con una routine che visualizza un messaggio dopo l'altro a intervalli generati da un loop interno di ritardo. Ma si tratta di una soluzione poco efficiente, in quanto il loop che crea l'intervallo fra un messaggio e l'altro occupa inutilmente il processore per tutta la durata del ritardo, non permettendogli (interrupt a parte) di compiere altre operazioni. Se invece la routine che risolve il problema adotta l'opzione sleep ogni volta che visualizza un messaggio, e imposta un tempo di sleep tale da permettere all'utente di leggere l'intero messaggio, il processore può effettuare altre operazioni durante la sua temporanea inattività, come ad esempio eseguire le istruzioni di MainLoop o altro. Quando la routine si risveglia, visualizza un altro messaggio e riprende a dormire in attesa di essere risvegliata nuovamente e così continua fino a quando non ha esaurito i messaggi da visualizzare. Ogni volta che si addormenta, il processore può dedicarsi ad altro. Quindi l'utente può per esempio scrivere un testo mentre sullo schermo, in un'opportuna finestra, continua ad apparire una serie di messaggi.

Nessuna delle variabili di gestione dei processi è accessibile direttamente: possono essere manipolate solo attraverso le routine che ora descriveremo.

InitProcesses

Funzione: Inizializza tutti i processi definiti, ma non li abilita.

Indirizzo: $C103

Parametri:
a numero di processi (1 - 20)
r0 puntatore alla tavola di definizione dei processi

Restituisce: r0 inalterato

Distrugge: a, x, y

Sinossi: InitProcesses copia dalla tavola di definizione gli indirizzi delle routine di servizio e gli intervalli di ripetizione, e imposta nello stato di congelato ogni processo. Per abilitare un processo l'applicazione deve chiamare la routine RestartProcess.

GEOS non è in grado di gestire più di 20 processi simultaneamente. Questa routine assegna a ogni processo una locazione che ne identifica lo stato. Nella versione 1.2 e 1.3 del Kernel queste locazioni sono disposte sequenzialmente a partire dall'indirizzo processFlags ($8719), e non possono essere più di 20. La prima locazione, processFlags + 0, è assegnata al processo 1, la seconda locazione, processFlags + 1, è assegnata al processo 2, e così via. Nell'appendice A sono riportate le costanti che individuano i possibili stati dei processi. L'applicazione non dovrebbe comunque avere necessità di accedere a processFlags, dal momento che il Kernel prevede tutte le routine necessarie alla gestione dei processi.

RestartProcess

Funzione: Sblocca e scongela un processo inizializzandone il timer.

Indirizzo: $C106

Parametri: x il numero del processo da attivare/riattivare, 0 per il primo processo

Restituisce: x inalterato

Distrugge: a

Sinossi: Abilita l'esecuzione di un processo a intervalli fissi predefiniti. Per prima cosa la routine sblocca e scongela il processo in modo da renderlo eseguibile. Subito dopo il timer viene aggiornato con il valore presente nella tavola di definizione e viene abilitata l'esecuzione del processo.

BlockProcess, UnblockProcess

Funzione: Blocca (Block) e sblocca (Unblock) l'esecuzione della routine di servizio di un processo.

Indirizzi: BlockProcess $C10C
UnblockProcess $C10F

Parametri: x numero del processo interessato al cambio di stato

Restituisce: x inalterato

Distrugge: a

Sinossi: BlockProcess: imposta un flag per indicare che la routine di servizio del processo non dev'essere eseguita. Nonostante questo, il timer di processo continua a essere decrementato, e viene riaggiornato se si azzera. Perché la routine associata al processo riprenda a essere eseguita con periodicità, bisogna che l'applicazione la sblocchi.

UnblockProcess: imposta a 0 il flag in maniera che la routine di servizio del processo torni a essere eseguita periodicamente.

FreezeProcess, UnfreezeProcess

Funzione: Congela (Freeze) e scongela (Unfreeze) il timer di un processo.

Indirizzi:

FreezeProcess	$C112
UnfreezeProcess	$C115

Parametri: x il numero del processo da congelare/scongelare

Restituisce: x inalterato

Distrugge: a

Sinossi: FreezeProcess: disattiva il funzionamento del timer associato al processo indicato. In questo modo la routine di servizio non può essere eseguita e il timer si mantiene al valore al quale è stato congelato.

UnfreezeProcess: riattiva il funzionamento del timer associato al processo indicato. In questo modo la routine di servizio torna a essere eseguita con periodicità.

Sleep

Funzione: Interrompe l'esecuzione della routine che ha eseguito l'istruzione jsr Sleep per un tempo determinato.

Indirizzo: $C199

Parametri: r0 indica l'intervallo di tempo, in sessantesimi di secondo, per il quale la routine deve rimanere in stato di sleep

Restituisce: Non definibile

Distrugge: Non definibile

Sinossi: Sleep salva l'indirizzo a cui cederà il controllo quando la routine sarà risvegliata, e forza l'esecuzione dell'istruzione rts, terminando così la routine nell'esatto punto dove avviene la chiamata jsr Sleep. Quindi se la routine 1 chiama la routine 2 e quest'ultima esegue la routine Sleep, GEOS salva un puntatore che indica il punto esatto nella routine 2 dove riprenderà l'esecuzione, e forza un rts in modo che il controllo ritorni alla routine 1. Questa, ripreso il controllo, si completa e ritorna a MainLoop. È a questo punto che MainLoop inizia a interagire con il modulo Sleep. Quando l'intervallo di sleep della routine 2 è terminato, GEOS cede il controllo all'indirizzo specificato dal puntatore precedentemente salvato, e la routine 2 termina, a sua volta, la propria esecuzione. Ma quando passa all'istruzione rts, il controllo non viene ceduto alla routine 2 che l'aveva chiamata, ma al modulo sleep presente in MainLoop che l'ha risvegliata. Questo è facilmente spiegabile. Quando la routine 2 si addormenta, il controllo viene ceduto alla routine 1 tramite l'istruzione rts. Routine 1 non può sapere se routine 2 le ha ceduto il controllo perché è stata forzata a farlo o perché la routine è stata realmente eseguita. Del resto la cosa le è indifferente, e riprende l'esecuzione. Dal momento che routine 2 è già ritornata a routine 1 una volta, non ha senso che lo rifaccia anche la seconda volta, quando termina "realmente" la propria esecuzione, e così il controllo viene ceduto al modulo sleep di MainLoop. Proprio per questo, è preferibile che una routine che al suo interno utilizza l'opzione sleep, sia stata originariamente chiamata da MainLoop: solo così, durante le successive chiamate a Sleep e alla fine della routine, il controllo viene ceduto alla routine che aveva fatto la prima chiamata, cioè MainLoop.

Comunque anche una subroutine può contenere una chiamata a Sleep, purché si tenga conto delle differenze che intervengono in questo caso. Vediamone un esempio.

```
CallingRoutine:
                ...
                codici
                ...
                jsr PrintLater
                ...
                codici
                ...
                rts

PrintLater:
                lda     #<N     ;passa il numero N di interrupt di attesa
                sta     r0L
                lda     #>N
                sta     r0H
                jsr     Sleep   ;esegue Sleep restituendo il codice alla routine
                                ;che l'aveva chiamata
                ...
Codici per stampare qualcosa in ritardo
                ...
                rts
```

In questo caso, i codici di PrintLater successivi all'istruzione jsr Sleep (che non devono dipendere dallo stato della macchina e non devono alterarlo) vengono eseguiti per effettuare la stampa N chiamate di interrupt dopo la chiamata, e nel frattempo la routine che ha fatto la chiamata è libera di continuare.

EnableProcess

Funzione: Forza l'esecuzione di un processo, qualunque sia lo stato in cui si trova.

Indirizzo: $C109

Parametri: x numero del processo da eseguire

Restituisce: x inalterato

Distrugge: a

Sinossi: Forza l'esecuzione del processo durante il successivo ciclo di MainLoop. Se il processo è bloccato e/o congelato, dopo la sua esecuzione forzata ritorna nello stato di bloccato e/o congelato. Se il processo è invece eseguibile, viene forzata la sua esecuzione e il timer torna al valore iniziale per riprendere a decrementarsi.

EnableProcess è utile per essere sicuri che un processo viene eseguito almeno una volta. Il processo potrebbe essere già eseguibile o in uno degli altri due stati. Questa routine non cambia assolutamente niente nello stato del processo, e si limita a forzarne l'esecuzione una volta.

EnableProcess è utile anche nel caso che l'applicazione preveda una propria routine di interrupt per gestire un dispositivo speciale, e che questa rilevi una condizione tale per cui nasca l'esigenza di sfruttare MainLoop per eseguire determinate routine. La distinzione importante, in questo caso, è che l'esigenza di eseguire particolari routine si scopre durante gli interrupt di sistema, ma le routine vengono eseguite durante MainLoop.

10 LIBRERIA DI FUNZIONI MATEMATICHE

Questo capitolo illustra le routine matematiche installate nel Kernel di GEOS. Questa libreria di routine prevede una varietà di utili funzioni matematiche, inclusa la moltiplicazione e la divisione fra word e byte. Eccone l'elenco.

DShiftLeft	Slitta una word a sinistra di n bit
DShiftRight	Slitta una word a destra di n bit
BBMult	Moltiplicazione fra due byte
BMult	Moltiplicazione fra un byte e una word
DMult	Moltiplicazione fra due word
Ddiv	Divisione fra due word
DSdiv	Divisione fra due word con segno
Dabs	Valore assoluto di una word
Dnegate	Negazione di una word complementata a due
DDec	Decrementazione di una word priva di segno
GetRandom	Genera una word casuale

DShiftLeft - Scorrimento a sinistra in doppia precisione

Funzione: Fa "slittare" i bit dell'operando di n posizioni a sinistra all'interno del byte. Il calcolo è: (operando * 2^n).

Indirizzo: $C15D

Parametri:
x indirizzo del registro in pagina zero (per esempio ldx #r1)
y numero di posizioni di cui i bit dell'operando devono slittare a sinistra

Restituisce: (x) il registro puntato da x viene slittato y volte a sinistra
a, x inalterati

Distrugge: y, $FF

Sinossi: DShiftLeft è una routine che calcola in doppia precisione. Effettua lo slittamento a sinistra dei bit contenuti nello pseudoregistro da 16 bit puntato da x. Lo slittamento avviene per un numero y di posizioni. L'equazione per questo calcolo è: (operando * 2^n). Tutti i flag di stato contenuti nel PSW rimangono inalterati.

DShiftRight - Scorrimento a destra in doppia precisione

Funzione: Fa "slittare" i bit dell'operando di n posizioni a destra all'interno del byte. Il calcolo è: (operando / 2^n).

Indirizzo: $C262

Parametri:
x indirizzo del registro in pagina zero (per esempio ldx #r1)
y numero di posizioni di cui i bit dell'operando devono slittare a destra

Restituisce:
(x) il registro puntato da x viene slittato y volte a destra
a, x inalterati

Distrugge: y, $FF

Sinossi: DShiftRight è una routine che calcola in doppia precisione. Effettua uno slittamento a destra dei bit contenuti nello pseudoregistro da 16 bit puntato da x. Lo slittamento avviene per un numero y di posizioni. L'equazione per questo calcolo è: (operando / 2^n). Tutti i flag di stato contenuti nel PSW rimangono inalterati.

BBMult - Moltiplicazione byte * byte

Funzione: Moltiplica fra loro due operandi privi di segno e memorizza il prodotto nel registro in pagina 0 puntato da x.

Indirizzo: $C160

Parametri:
x indirizzo dello pseudoregistro destinazione in pagina 0 (per esempio ldx #r1)
y indirizzo dello pseudoregistro sorgente in pagina 0 (per esempio ldy #r2)

Restituisce:
(x) il registro puntato da x contiene il risultato da 16 bit
x, y inalterato

Distrugge: a, r7, r8

Sinossi: BBMult moltiplica due byte fra di loro e memorizza il prodotto nella destinazione. Se x contiene l'indirizzo di r5, il byte basso del registro r5 (r5L) viene usato come primo operando della moltiplicazione. Questo viene moltiplicato per il byte individuato dal registro puntato da y, e il risultato dell'operazione, che occupa 16 bit, viene memorizzato nei due byte del registro r5.

BMult - Moltiplicazione word * byte

Funzione: Moltiplica fra loro una word priva di segno e un byte privo di segno, e memorizza il prodotto nella word puntata da x.

Indirizzo: $C163

Parametri:
x indirizzo dello pseudoregistro destinazione in pagina 0 (per esempio ldx #r1)
y indirizzo dello pseudoregistro sorgente in pagina 0 (per esempio ldy #r2)

Restituisce:
(x) il registro da 16 bit puntato da x contiene il risultato della moltiplicazione
(y) il byte alto del registro sorgente puntato da y viene azzerato
x, y inalterati

Distrugge: a, r6, r8

Sinossi: BMult moltiplica un byte con una word e memorizza il risultato nella word. Il byte operando sorgente è puntato da y. La word operando destinazione è puntata da x. Tutti i flag di stato contenuti nel PSW rimangono inalterati.

DMult - Moltiplicazione in doppia precisione

Funzione: Moltiplica fra loro due word prive di segno.

Indirizzo: $C166

Parametri:
- x — indirizzo dello pseudoregistro destinazione in pagina zero (per esempio ldx #r1)
- y — indirizzo dello pseudoregistro sorgente in pagina 0 (per esempio ldx #r2)

Restituisce:
- (x) — il registro puntato da x contiene il risultato da 16 bit dell'operazione
- (y) — il registro puntato da y rimane inalterato
- x, y — inalterati

Distrugge: a, r6 - r8

Sinossi: DMult moltiplica la word sorgente per la word destinazione e memorizza il prodotto nella word destinazione. La word operando sorgente è puntata da y. La word operando destinazione è puntata da x. Tutti i flag di stato contenuti nel PSW rimangono inalterati.

Ddiv - Divisione in doppia precisione

Funzione: Divide una word destinazione priva di segno per una word sorgente e memorizza il quoziente nella word destinazione. Il resto della divisione è memorizzato in r8.

Indirizzo: $C169

Parametri:

x	indirizzo dello pseudoregistro destinazione in pagina zero (per esempio ldx #r1)
y	indirizzo dello pseudoregistro sorgente in pagina zero (per esempio ldx #r2)

Restituisce:

(x)	il registro puntato da x contiene il risultato da 16 bit dell'operazione
r8	il resto dell'operazione
(y)	il registro puntato da y non viene alterato
x, y	inalterati

Distrugge: a, r9

Sinossi: Ddiv divide una word destinazione per una word sorgente e memorizza il risultato nella word destinazione. Il resto della divisione è memorizzato in r8. La word operando sorgente è puntata da y. La word operando destinazione è puntata da x. Tutti i flag di stato del PSW restano inalterati. Se per esempio l'operazione è 4 / 3, la word destinazione riceve il quoziente intero 1 ed r8 riceve il resto della divisione, cioè 1.

DSdiv - Divisione in doppia precisione con segno

Funzione: Divide una word destinazione con segno per una word sorgente con segno e memorizza il quoziente nella word destinazione. Il resto della divisione è memorizzato in r8.

Indirizzo: $C16C

Parametri:
- x — indirizzo dello pseudoregistro destinazione in pagina 0 (per esempio ldx #r1)
- y — indirizzo dello pseudoregistro sorgente in pagina 0 (per esempio ldy #r2)

Restituisce:
- (x) — il registro puntato da x contiene il risultato da 16 bit dell'operazione
- r8 — contiene il resto della divisione
- (y) — il registro puntato da y rimane inalterato
- x, y — inalterati

Distrugge: a, r9

Sinossi: DSdiv divide una word destinazione con segno per una word sorgente con segno, e memorizza il quoziente nella word destinazione. Il resto della divisione viene memorizzato in r8. La word operando sorgente è puntata da y. La word operando destinazione è puntata da x. Tutti i flag di stato del PSW sono inalterati. Il calcolo aritmetico viene eseguito con la tecnica del complemento a due. Per esempio la divisione 4 / –3 produce il quoziente –1 memorizzato nella word destinazione e il resto 1 memorizzato in r8.

Dabs - Valore assoluto in doppia precisione

Funzione: Calcola il valore assoluto di una word in complemento a due.

Indirizzo: $C16F

Parametri: x indirizzo dello pseudoregistro in pagina 0 (per esempio ldx #r1)

Restituisce: (x) il registro puntato da x riceve il valore assoluto in complemento a due dell'operando in esso originariamente memorizzato

Distrugge: a

Sinossi: Dabs calcola il valore assoluto della word memorizzata nel registro puntato da x. Tutti i flag di stato del PSW sono inalterati. Il calcolo aritmetico viene effettuato in complemento a due.

Dnegate - Negazione in doppia precisione con segno

Funzione: Nega una word in complemento a due.

Indirizzo: $C172

Parametri: x indirizzo della word pseudoregistro in pagina 0 (per esempio ldx #r1)

Restituisce: (x) la word nel registro puntato da x riceve il risultato dell'operazione
x inalterato

Distrugge: a, y

Sinossi: Dnegate nega la word in complemento a due memorizzata nel registro puntato da x (cioè ne cambia il segno). Tutti i flag di stato restano inalterati. Il calcolo aritmetico viene effettuato in complemento a due.

Ddec - Decrementazione di una word priva di segno

Funzione: Decrementa una word priva di segno.

Indirizzo: $C175

Parametri: x indirizzo della word pseudoregistro in pagina 0 (per esempio ldx #r1)

Restituisce: (x) la word priva di segno nel registro puntato da x viene decrementata
x inalterato

Distrugge: a

Sinossi: Ddec decrementa la word priva di segno memorizzata nel registro puntato da x. Tutti i flag di stato restano inalterati. Il calcolo aritmetico è effettuato per valori positivi (non in complemento a due) su 16 bit.

GetRandom

Funzione: Genera un numero da 16 bit pseudocasuale.

Indirizzo: $C187

Parametri: Nessuno

Restituisce: random contiene un nuovo numero da 16 bit

Distrugge: a

Sinossi: GetRandom produce un nuovo numero da 16 bit pseudocasuale nella variabile globale random. L'algoritmo di generazione è il seguente:

numero nuovo = (numero precedente+1)*2 mod 65521

Il numero casuale generato è sempre minore di 65221, e ha più o meno un'uniforme distribuzione fra 0 e 65221.

11 LIBRERIA DI ROUTINE DI UTILITÀ GENERALE

Questa libreria di routine è stata realizzata per la manipolazione delle stringhe, gli spostamenti di gruppi di dati, le inizializzazioni e altre funzioni di utilità generale. Sono routine impiegate anche dal Kernel di GEOS per le operazioni interne. Eccone l'elenco schematico, con una breve spiegazione.

CopyString	Copia una stringa a terminazione nulla in un'altra
CopyFString	Copia una stringa di lunghezza fissa
CmpString	Confronta una stringa a terminazione nulla con un'altra
CmpFString	Confronta due stringhe di lunghezza fissa
Panic	Routine eseguita se il processore incontra l'istruzione BRK
MoveData	Muove un'area di memoria di dimensioni arbitrarie in un'altra
ClearRam	Azzera una zona di memoria di dimensioni arbitrarie
FillRam	Riempie una particolare zona di memoria con il valore di un byte
InitRam	Aggiorna la RAM leggendo una tavola d'inizializzazione
CallRoutine	Esegue la routine con l'indirizzo specificato
GetSerialNumber	Restituisce il numero di serie del Kernel di GEOS
ToBasic	Passa il controllo del processore da GEOS al Basic del C-64
FirstInit	Inizializza l'intero sistema
CRC	Esegue un checksum (controllo di somma)
DoInlineReturn	Viene eseguita al termine di una routine inline

CopyString

Funzione: Copia una stringa a terminazione nulla da una zona di memoria a un'altra.

Indirizzo: $C265

Parametri:
x indirizzo del registro in pagina 0 che individua la stringa sorgente
y indirizzo del registro in pagina 0 che individua la stringa destinazione

Restituisce: ((y)) la stringa copiata alla locazione di memoria puntata dal registro che a sua volta è puntato da y

Distrugge: a, x, y

Sinossi: CopyString copia una stringa a terminazione nulla di lunghezza arbitraria da un'area di memoria a un'altra. È ovvio che la stringa non deve contenere al suo interno dei byte a zero.

CopyFString

Funzione: Copia un numero dato di byte da una zona di memoria a un'altra.

Indirizzo: $C268

Parametri:
- x indirizzo dello pseudoregistro in pagina 0 contenente il puntatore alla stringa sorgente
- y indirizzo dello pseudoregistro in pagina 0 contenente il puntatore alla stringa destinazione
- a numero di byte da copiare

Restituisce: ((y)) la stringa copiata alla locazione puntata dal registro in pagina 0 che a sua volta è puntato da y

Distrugge: a, x, y

Sinossi: CopyFString copia una stringa di lunghezza fissa da un'area di memoria a un'altra. A differenza di CopyString, la stringa da copiare può contenere byte a zero.

CmpString

Funzione: Confronta fra loro due stringhe a terminazione nulla.

Indirizzo: $C26B

Parametri:
- x indirizzo del registro in pagina 0 contenente il puntatore alla stringa sorgente
- y indirizzo del registro in pagina 0 contenente il puntatore alla stringa destinazione

Restituisce:
- Z flag di zero impostato a 1 se le stringhe sono uguali
- N flag di segno impostato a 1 se il primo byte della stringa sorgente diverso dal corrispondente nella stringa destinazione, è più piccolo di quest'ultimo

Distrugge: a, x, y

Sinossi: CmpString confronta una stringa a terminazione nulla di lunghezza arbitraria con un'altra. Dal momento che il segno viene trascurato nel confronto, se le stringhe sono diverse il flag N nel PSW viene impostato a 1 nel caso che il byte della stringa sorgente diverso dal corrispondente della stringa destinazione sia il minore dei due.

CmpFString

Funzione: Confronta un assegnato numero di byte contenuti in un'area di memoria con i byte contenuti in un'altra area di memoria.

Indirizzo: $C26E

Parametri:
- x indirizzo del registro in pagina zero contenente il puntatore alla stringa sorgente
- y indirizzo del registro in pagina zero contenente il puntatore alla stringa destinazione
- a numero di byte da confrontare

Restituisce:
- Z flag di zero a 1 se le stringhe sono uguali
- N flag di segno a 1 se il primo byte della stringa sorgente diverso dal corrispondente nella stringa destinazione, è più piccolo di quest'ultimo

Distrugge: a, x, y

Sinossi: CmpFString confronta una stringa di byte di lunghezza fissa da un'area di memoria a un'altra. Dal momento che il numero di byte interessati dal confronto viene indicato nel registro a, il massimo numero di byte confrontabili è 256. Differentemente da CmpString che interrompe il confronto quando incontra un byte a zero nella stringa sorgente, CmpFString consente la presenza di zeri nelle stringhe di byte da confrontare. Dal momento che il segno viene trascurato nel confronto, se le stringhe sono diverse il flag N nel PSW viene impostato a 1 nel caso che il byte della stringa sorgente diverso dal corrispondente della stringa destinazione, sia il minore dei due.

Panic

Funzione: È il modo standard d'interpretare l'istruzione BRK. Apre un box di dialogo e visualizza al suo interno l'indirizzo al quale il processore ha incontrato l'istruzione BRK.

Indirizzo: $C2C2

Chiamata da: Un errore nel codice

Chiama: DoDlgBox

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, x, y, r0 - r15, e a volte molto di più

Sinossi: Quando il processore incontra un'istruzione BRK, GEOS blocca le chiamate di interrupt e apre sullo schermo un box di dialogo in cui si trova l'indirizzo dell'istruzione BRK che ha causato l'interruzione. È un'utile tecnica per correggere i codici errati che cedono inavvertitamente il controllo del processore a una zona di memoria non significativa. Di solito accade che il processore esegue istruzioni che di fatto non sono altro che strutture di dati o byte casuali, e fra queste è molto facile che si trovi un byte di valore zero. Nel momento in cui il processore tenta di eseguire l'istruzione BRK che ha codice 0, viene eseguita la routine Panic. Questa visualizza un box di dialogo che contiene il messaggio "System error near $xxxx", dove $xxxx è l'indirizzo dell'istruzione BRK incontrata. Quando GEOS si trova costretto a eseguire la routine Panic, il computer diventa completamente inattivo e bisogna spegnerlo per poterlo usare di nuovo.

Se l'applicazione deve gestire l'interruzione BRK con una propria routine, deve memorizzarne l'indirizzo del vettore BRKVector.

MoveData, i_MoveData

Funzione: Muove un blocco di memoria di dimensioni arbitrarie.

Indirizzi:
MoveData $C17E
i_MoveData $C1B7

Parametri:
Normali
r0 indirizzo sorgente del blocco
r1 indirizzo destinazione del blocco
r2 numero di byte da muovere

Inline
.word indirizzo sorgente del blocco
.word indirizzo destinazione del blocco
.word numero di byte da muovere

Restituisce: Il blocco di dati è stato trasferito

Distrugge: a, y, r0, r1, r2

Sinossi: MoveData muove un blocco di dati di lunghezza arbitraria da un'area di memoria a un'altra. L'area sorgente e l'area destinazione possono anche sovrapporsi.

ClearRam

Funzione: Azzera una specifica zona di memoria.

Indirizzo: $C178

Parametri: r0 numero di byte da azzerare
r1 indirizzo al quale iniziare l'azzeramento

Restituisce: Niente

Distrugge: a, y, r0, r1, r2L

Sinossi: Azzera la zona di memoria specificata.

FillRam, i_FillRam

Funzione: Riempie una zona di memoria RAM con il contenuto di un particolare byte.

Indirizzo:

FillRam	$C17B
i_FillRam	$C1B4

Parametri:

Normali

r0 numero di byte da riempire
r1 indirizzo del primo byte da aggiornare con il valore del byte specificato
r2L valore con il quale riempire l'area di memoria specificata

Inline

.word numero di byte da riempire
.word indirizzo al quale iniziare a riempire la memoria
.byte il valore da memorizzare nell'area di memoria

Restituisce: Niente

Distrugge: a, y, r0, r1, r2L

Sinossi: Aggiorna tutti i byte compresi nell'area indicata con il valore specificato in r2L. Questa routine è utile per impostare un'area di memoria con un valore diverso da zero.

InitRam

Funzione: Costituisce un semplice ed efficace sistema per inizializzare aree di memoria separate.

Indirizzo: $C181

Parametri: r0 indirizzo della tavola d'inizializzazione alla quale la routine fa riferimento

Restituisce: L'area o le aree di memoria inizializzate

Distrugge: a, x, y, r0, r1

Sinossi: InitRam legge una tavola che indica come inizializzare un'area di memoria RAM. È particolarmente utile in due casi.

1) Si desidera inizializzare una zona di memoria rapidamente e in modo compatto, e questa inizializzazione dev'essere effettuata con una certa frequenza. Per esempio può trattarsi di dover riportare un'applicazione allo stato di default.

2) Si desidera inizializzare un insieme di piccoli segmenti di RAM non continui. InitRam diventa particolarmente efficiente quando l'inizializzazione è del tipo: «due byte qui, tre byte là».

La tavola di definizione, puntata da r0, contiene una o più entrate (gruppo di dati), ognuna in questa forma:

```
.word                                  La prima word specifica la
                                       locazione della RAM che l'entrata
                                       corrente deve inizializzare
.byte                                  Numero di byte da inizializzare
.byte   value1, value2, ..., valueN    Valori da riportare a partire
                                       dall'indirizzo specificato
```

Nella tavola possono essere presenti molte di queste "entrate". Se all'inizio di una nuova entrata il valore della word è 0, la routine interpreta lo 0 come terminatore della tavola. Riportiamo qui di seguito un esempio di tavola di definizione composta da due entrate.

Semplice tavola d'inizializzazione

```
.word       FirstLocation  ;locazione in memoria da cui
                           ;cominciare l'inizializzazione
                           ;con la prima entrata
.byte       3              ;inizializza tre byte
.byte       4, 5, 6        ;valori da memorizzare
                           ;nei 3 byte allocati a partire
                           ;da FirstLocation

.word       SecondLocation ;locazione in memoria da cui
                           ;cominciare l'inizializzazione
                           ;con la seconda entrata
.byte       10             ;inizializza 10 byte
.word       30, $FFE5      ;si possono miscelare word
.byte       0, 0, 2, 3, 1  ;e byte, l'importante e' che
                           ;i byte totali siano 10
.word       0

.word       0              ;se la word contiene il valore 0
                           ;al posto di un indirizzo,
                           ;all'inizio dell'entrata,
                           ;la routine rileva che
                           ;la tavola e' terminata
```

CallRoutine

Funzione: Effettua un salto (jmp) indiretto alla routine il cui indirizzo è memorizzato in x e in a.

Indirizzo: $C1D8

Parametri:
a byte basso dell'indirizzo della routine da chiamare
x byte alto dell'indirizzo della routine da chiamare

Restituisce: Dipende dalla routine chiamata. CallRoutine non restituisce niente

Distrugge: Dipende dalla routine chiamata. CallRoutine non distrugge niente

Sinossi: CallRoutine effettua una chiamata indiretta all'indirizzo passato in a e in x. Se a e x valgono 0, nessuna routine viene chiamata e il risultato di CallRoutine è un semplice rts.

GetSerialNumber

Funzione: Restituisce il numero di serie del Kernel di GEOS.

Indirizzo: $C196

Parametri: Nessuno

Restituisce: r0 il numero di serie

Distrugge: a

Sinossi: GetSerialNumber restituisce il numero di serie del Kernel di GEOS correntemente in memoria. Questo valore è memorizzato nei codici del Kernel. Le applicazioni possono crearsi una chiave di funzionamento, memorizzando su disco il numero di serie del Kernel da cui sono state gestite la prima volta. In questo modo l'applicazione diventa per così dire "Kernel dipendente", e si rifiuta di funzionare se il Kernel in memoria ha un numero di serie diverso da quello registrato in origine come chiave. L'applicazione funzionerà esclusivamente sotto il Kernel dell'utente che l'ha comprata. Ogni Kernel di GEOS ha un numero di serie autonomo e diverso da tutti gli altri.

ToBasic

Funzione: Passa il controllo da GEOS al Basic del C-64. È possibile indicare un comando Basic, o una serie di comandi, in modo che vengano immediatamente eseguiti. La routine è anche in grado di caricare in memoria un file tramite il turbo di GEOS.

Indirizzo: $C241

Parametri:
- r7 inizio delle variabili Basic in memoria
- r0 puntatore alla stringa a terminazione nulla che contiene i comandi Basic da impartire automaticamente
- r5 se r5H è diverso da 0, la routine utilizza l'indirizzo contenuto in r5 per puntare il File Entry di un file che deve già essere stato predisposto in memoria, accede al File Entry e carica il file in memoria tramite i codici turbo

Restituisce: Niente

Distrugge: In linea di massima tutti i registri

Sinossi: ToBasic è utilizzata per trasferire il controllo al Basic del C-64. Attiva il Kernel del C-64, lo spazio di I/O e la ROM del Basic, e il sistema viene nuovamente inizializzato. r7 deve indicare dove inizia lo spazio per le variabili Basic. Questo indirizzo diminuito di 2 individua il primo byte dello spazio che conterrà il file da caricare eventualmente in memoria.

Se il byte alto r5H è diverso da zero, r5 indica un buffer che dovrebbe contenere il File Entry di un file da caricare in memoria. La routine accede ai dati contenuti nel File Entry e carica il file in memoria a partire dall'indirizzo (r7 – 2), impiegando il turbo di GEOS. Attenzione: la routine non si accerta che le dimensioni del file siano tali da non oltrepassare l'indirizzo $7FFF, limite al di là del quale inizia l'area dati del Kernel di GEOS. È necessario quindi che l'applicazione verifichi che il file non sia di dimensioni eccessive. Se r5H vale zero, la routine non carica in memoria alcun file.

r0 indica l'indirizzo della stringa a terminazione nulla che contiene i comandi diretti che il Basic deve eseguire appena riceve il controllo da GEOS. I comandi devono essere separati l'uno dall'altro dal carattere CR ($13), e la stringa non deve contenere più di 40 caratteri. Ogni comando dev'essere memorizzato nella stringa esattamente come verrebbe battuto

da tastiera, e non deve quindi includere alcun token Basic. L'interprete Basic, appena ricevuto il controllo, esegue la serie di comandi sequenzialmente. La stringa deve terminare con un carattere CR e uno zero che ne individua la fine. Se non si desidera impartire alcun comando, si deve passare una stringa nulla, cioè con il primo carattere uguale a zero.

Per esempio, se l'applicazione deve caricare un file non compatibile GEOS e mandarlo in esecuzione, deve poter verificare se le sue dimensioni in memoria interferiscono con il Kernel di GEOS. Se il file non oltrepassa il limite $7FFF, l'applicazione può eseguire la routine ToBasic indicando in r5 l'indirizzo in memoria del File Entry del file da caricare. In questo modo la routine carica il file in memoria tramite il turbo, e successivamente cede il controllo al Basic ordinando di eseguire la stringa di comandi contenuta in r7. Se invece l'applicazione verifica che il file da caricare in memoria è troppo grande, non fa altro che aggiungere nella stringa di comandi l'istruzione LOAD "Nome File", 8, 1 in maniera che la prima operazione che eseguirà il Basic, sarà il caricamento del file in memoria. Il comando successivo potrebbe essere l'istruzione RUN o l'istruzione SYS (xxxxx), per mandare in esecuzione il file.

FirstInit

Funzione: FirstInit effettua una completa partenza "a caldo" del sistema.

Indirizzo: $C271

Parametri: Nessuno

Restituisce: Il sistema nella sua configurazione "a caldo" (Warm Start)

Distrugge: a, y, r0, r1, r2L

Sinossi: FirstInit è la routine chiamata per inizializzare il sistema. Nel capitolo 20 è illustrata la tavola che riporta la configurazione di Warm Start (partenza a caldo) effettuata da FirstInit.

CRC

Funzione: CRC effettua il checksum (controllo di somma) su una particolare area di dati.

Indirizzo: $C20E

Parametri:
r0 puntatore all'inizio del gruppo di dati
r1 numero di byte da considerare nell'operazione di checksum

Restituisce: r2 risultato del checksum

Distrugge: a, x, y, r0, r1, r3L

Sinossi: CRC effettua un lungo e complesso controllo di somma nell'area di memoria specificata. Per quanto venga chiamato "controllo di somma", l'algoritmo impiegato da CRC non è una somma. Si tratta di un laborioso calcolo che viene effettuato sull'intera struttura di dati indicata, senza seguire la sequenza dei byte. Il controllo è comunque rigorosamente logico e determina in qualunque caso un risultato diverso se è stato variato anche un solo byte all'interno della struttura di dati interessata.

Se si desidera proteggere un programma da alterazioni su disco, è sufficiente inserirvi all'interno la routine che effettua il controllo di somma chiamando CRC, e, con un programma a parte, chiamare CRC per ottenere il controllo di somma sull'intero programma, lasciando ovviamente fuori dal controllo la word che andrà a contenere il valore ottenuto. In questo modo si è sicuri di identificare subito qualunque alterazione nei codici del programma.

DoInlineReturn

Funzione: Esegue il ritorno da una subroutine per le routine inline. Il Kernel di GEOS la esegue per diverse routine inline, e anche le applicazioni possono impiegarla per gestire al proprio interno routine inline.

Indirizzo: $C2A4

Parametri:

a	numero di byte parametri + 1
stack	deve contenere come ultimo byte il PSW
returnAddress	indirizzo primo byte parametro – 1

Restituisce: Niente

Distrugge: a

Sinossi: Vediamo come una routine può trarre vantaggio dal passagio dei parametri inline e come viene impiegata DoInlineReturn. La routine inline, appena ricevuto il controllo, deve memorizzare nella variabile returnAddress l'indirizzo di ritorno da subroutine contenuto nello stack. Questo indirizzo non punta però al primo byte dopo la chiamata jsr Routine (Routine è la nostra ipotetica routine), ma alla locazione di memoria appena precedente. Supponendo che l'istruzione jsr Routine si trovi all'indirizzo $1000, sullo stack viene memorizzata la word $1002 che, incrementata di uno, costituisce l'indirizzo di ritorno della subroutine. Una volta che Routine ha salvato, svuotando lo stack, l'indirizzo di ritorno – 1 nella variabile returnAddress, procede alla lettura dei parametri che seguono la chiamata jsr Routine. Per accedervi impiega la variabile returnAddress e l'indice y inizialmente a 1, e non a 0 come si potrebbe pensare. Routine conosce esattamente il numero di byte parametri che deve leggere e procede incrementando per ognuno l'indice y.

Quando Routine ha prelevato tutti i parametri che le competono, inizia lo svolgimento della sua funzione. Compiute tutte le operazioni, Routine deve restituire il controllo alla routine di livello superiore che l'ha mandata in esecuzione. Per far questo, deve inizialmente salvare sullo stack il PSW eseguendo l'istruzione php. Quest'operazione è necessaria perché DoInline-Return potrebbe alterare alcuni flag che Routine restituisce come parametri di output. Successivamente deve memorizzare nell'accumulatore il numero di byte parametri + 1 che sono stati passati, ed eseguire l'istruzione

jmp DoInlineReturn (nota bene: la chiamata a DoInlineReturn dev'essere assolutamente eseguita tramite l'istruzione jmp). A questo punto è già stato effettuato il ritorno da subroutine.

Supponiamo che Routine debba ricevere soltanto due byte parametri e vediamo come dev'essere organizzata.

```
Routine:

    Pop     returnAddress       ;preleva la word di ritorno dallo stack
                                ;e la memorizza in returnAddress
    ldy     #01                 ;aggiorna l'indice a 1
    lda     (returnAddress), y  ;preleva il primo byte parametro
    sta     dato1               ;e lo memorizza
    iny                         ;incrementa l'indice
    lda     (returnAddress), y  ;preleva il secondo byte parametro
    sta     dato2               ;e lo memorizza
    ...
    ...                         ;la routine svolge i suoi compiti
    ...
    php                         ;salva il PSW
    lda     #(2 + 1)            ;numero di byte parametri + 1
    jmp     DoInlineReturn
```

12 I BOX DI DIALOGO

I box di dialogo appaiono come un rettangolo sullo schermo all'interno del quale possono trovarsi testi, icone e stringhe, manipolabili in diversi modi. I box di dialogo vengono utilizzati dalle applicazioni per segnalare all'utente condizioni d'errore, avvisarlo della possibilità che si verifichino situazioni indesiderate, attendere frasi in input dall'utente, creare una lista di nomi di file per operare una selezione o soddisfare altre esigenze di dialogo fra l'utente e l'applicazione. Tra le strutture di box di dialogo impiegate più frequentemente, alcune sono già disponibili nel Kernel di GEOS. Usati nell'ambito delle funzioni realizzate dal programmatore, i box di dialogo costituiscono un'interfaccia utente semplice, compatta e molto flessibile.

Un box di dialogo può essere aperto sullo schermo in qualsiasi momento. Si può considerare come una piccola applicazione che viene eseguita nell'ambiente che le è proprio. Non altera in alcun modo l'esecuzione dell'applicazione corrente e non cambia le sue variabili (tranne nel caso in cui il programmatore abbia previsto una routine di servizio associata che volutamente agisca sulle variabili dell'applicazione). Chiamando l'apertura di un box di dialogo, la maggior parte dello stato della macchina viene provvisoriamente salvato. Tutte le variabili del Kernel, i vettori, le strutture dei menu e delle icone vengono preventivamente salvati. Quindi i box di dialogo possono essere anche molto elaborati, dal momento che non corrono il rischio d'intaccare lo stato del sistema. Un'eccezione è costituita dagli pseudoregistri r0H - r15 che non vengono salvati. Il metodo per ripristinare la parte di schermo coperta dal box di dialogo (BD), nel caso che non sia abilitato il buffer di schermo, è descritto in questo stesso capitolo.

Per aprire un box di dialogo si deve chiamare la routine DoDlgBox, mentre per chiuderlo e restituire il controllo all'applicazione si deve eseguire la routine RstrFrmDialogue. Tutte le variabili necessarie per gestire un box di dialogo sono riassunte in una tavola di definizione semplice, ma molto funzionale. Questa tavola specifica le dimensioni del BD e le sue funzioni. È composta da una serie di byte

comando e dai byte parametri associati. I byte comando di un BD possono indicare icone da visualizzare o comandi (normalmente per visualizzare testi) da eseguire mentre il BD è aperto. I byte parametri di un BD specificano informazioni come la locazione del BD, le sue dimensioni e le stringhe da visualizzare.

Lungo la tavola di definizione del BD, uno 0 al posto di un byte comando indica la fine della tavola.

Le icone e i comandi per gestire i box di dialogo

Il Kernel di GEOS supporta al suo interno un set di icone residenti, realizzate per essere utilizzate all'interno dei BD. Le icone dei BD rappresentano un metodo semplice per rispondere a domande o considerazioni dell'applicazione. Quando l'utente preme il pulsante del mouse su una delle icone visualizzate, normalmente il BD si cancella, il numero dell'icona selezionata viene restituito in r0L e viene automaticamente eseguita la routine RstrFrmDialogue. L'applicazione che aveva aperto il BD deve poi leggere il contenuto di r0L e operare di conseguenza, di solito chiamando la routine di servizio associata all'icona selezionata. Sono disponibili in GEOS le icone per i BD che indicano YES, NO, OK, OPEN e DISK.

I comandi dei BD sono utili per chiamare routine arbitrarie, per visualizzare stringhe di testo, per ricevere in input stringhe di testo, per visualizzare un box all'interno del quale effettuare lo scroll dei nomi di alcuni file, per visualizzare un'icona definita dall'utente e per definire un vettore che punta a una routine da eseguire se l'utente preme il pulsante del mouse in un'area del box di dialogo diversa da un'icona. I comandi per i BD sono composti da un byte che specifica il numero del comando ed eventualmente alcuni byte parametri.

La struttura dei box di dialogo

La prima informazione in una tavola di definizione di un BD è un byte comando che ne definisce la posizione. Questo comando può specificare una posizione di default per il BD, DEF_DB_POS, o indicare una posizione definita dall'utente, SET_DB_POS, nel qual caso devono seguire le informazioni necessarie perché GEOS possa visualizzare i BD di dimensioni non standard.

Il byte di comando che definisce la posizione del BD è OR'ed, con il numero di una matrice grafica di sistema con la quale riempire l'ombra del BD. Il box ombra del BD è un rettangolo delle stesse dimensioni del BD ed è riempito con un'opportuna matrice grafica di sistema. Il box ombra appare sotto il BD ed è spostato rispetto al BD di 8 bit a sinistra e 8 bit verso il basso. Se il numero della matrice grafica corrisponde a 0 il box ombra non è richiesto. È senz'altro più facile farsi un'idea precisa di un BD associato al suo box ombra tramite un esempio, piuttosto che con lunghe descrizioni.

Nel corso del capitolo, nell'esempio OpenBox, viene descritta a questo scopo una figura con un BD completo di box ombra.

Le due forme del byte comando di posizione, che indicano rispettivamente un BD di dimensioni di default e un BD di dimensioni definite dall'applicazione, sono:

```
.byte   DEF_DB_POS|pattern      .byte   SET_DB_POS|pattern
                                .byte   latoSuperiore          ;(0 - 199)
                                .byte   latoInferiore          ;(0 - 199)
                                .word   latoSinistro           ;(0 - 319)
                                .word   latoDestro             ;(0 - 319)
```

I comandi di posizione

Dopo il byte, o il gruppo di byte, che definisce la posizione e le dimensioni del BD, possono seguire sia byte icona che byte comando. L'icona OK è la più frequente. Il byte che specifica l'icona OK è seguito da due byte che definiscono la posizione orizzontale e verticale come offset relativi all'angolo sinistro in alto del BD. Il primo è la coordinata x dell'icona misurata in byte (0 - 39); il secondo è la coordinata y dell'icona misurata in pixel (0 - 199). La definizione dell'icona OK deve seguire questa traccia

```
.byte     OK            ;costante che identifica l'icona OK
.byte     xOffset       ;(0 - 39  byte)
.byte     yOffset       ;(0 - 199 pixel)
```

Il byte icona OK

Quando un BD è attivato e l'utente seleziona una delle icone disponibili, il BD scompare e il sistema restituisce in r0L il numero dell'icona selezionata. L'applicazione, accedendo a r0L, può eseguire le operazioni legate alla scelta dell'utente. All'interno di un BD non possono essere visualizzate più di otto icone.

La tavola riportata nella pagina successiva elenca i comandi per gestire le icone all'interno di un BD.

Icone per i box di dialogo

Icona	Valore	Esempio
OK	1	.byte OK .byte xpos_in_byte .byte ypos_in_pixel
CANCEL	2	.byte CANCEL .byte xpos_in_byte .byte ypos_in_pixel
YES	3	.byte YES .byte xpos_in_byte .byte ypos_in_pixel
NO	4	.byte NO .byte xpos_in_byte .byte ypos_in_pixel
OPEN	5	.byte OPEN .byte xpos_in_byte .byte ypos_in_pixel
DISK	6	.byte DISK .byte xpos_in_byte .byte ypos_in_pixel
FUTURE1	7	.byte FUTURE1 .byte xpos_in_byte .byte ypos_in_pixel
FUTURE2	8	.byte FUTURE2 .byte xpos_in_byte .byte ypos_in_pixel
FUTURE3	9	.byte FUTURE3 .byte xpos_in_byte .byte ypos_in_pixel
FUTURE4	10	.byte FUTURE4 .byte xpos_in_byte .byte ypos_in_pixel

I comandi dei box di dialogo

Numerosi comandi sono stati previsti per conferire autonomia e flessibilità ai BD. Molti riguardano la visualizzazione di testi all'interno dei BD. Per esempio, il comando DBTXTSTR (il cui valore è 11) è seguito da due byte posizione e da una word che punta a una stringa di testo da visualizzare. Quando questo comando è impiegato all'interno di un BD, la posizione in cui iniziare a visualizzare il testo (coordinate x e y) si riferisce all'angolo sinistro in alto dello schermo. Le due coordinate sono misurate in pixel; la y si misura dal lato superiore del BD alla linea di base della stringa da visualizzare, mentre la x si misura dal lato sinistro del BD al lato sinistro del primo carattere della stringa. Le stringhe da visualizzare non possono avere una distanza orizzontale superiore a 256 pixel dal lato sinistro del BD.

Per passare i parametri ai comandi si possono impiegare i registri da r5 a r10, ed r15. Vediamo ora i comandi disponibili. Successivamente analizzeremo nei dettagli quelli più complessi.

DBTXTSTR - Dialog Box Text String (Stringa di testo per BD)

Codice del comando: 11

L'esempio che segue mostra la struttura del comando. DBTXTSTR visualizza la stringa di testo puntata da TextPtr, posizionandola all'interno del BD secondo le due coordinate xpos_offset e ypos_offset espresse in pixel e riferite all'angolo sinistro in alto del BD. TextPtr contiene l'indirizzo della stringa a terminazione nulla da visualizzare.

```
.byte   DBTXTSTR      ;byte comando
.byte   xpos_offset   ;offset dal lato sinistro del BD espresso in pixel (0 - 256)
.byte   ypos_offset   ;offset dal lato superiore del BD espresso in pixel (0 - 199)
.word   TextPtr       ;puntatore alla stringa a terminazione nulla
```

DBVARSTR - Dialog Box Variable Text String (Stringa di testo variabile per BD)

Codice del comando: 12

L'esempio che segue mostra la struttura del comando. DBVARSTR, a differenza di DBTXTSTR, visualizza la stringa a terminazione nulla puntata dal registro il cui indirizzo è contenuto in regAddr. Se regAddr è uguale a r5 ($0C), indica il registro r5. La stringa da visualizzare può essere diversa a ogni apparizione del BD secondo le esigenze dell'applicazione. La stringa è visualizzata all'interno del BD nella posizione indicata dalle due coordinate xpos_offset e ypos_offset, espresse in pixel e riferite all'angolo sinistro in alto del BD.

```
.byte   DBVARSTR      ;byte comando
.byte   xpos_offset   ;offset dal lato sinistro del BD espresso in pixel (0 - 256)
.byte   ypos_offset   ;offset dal lato superiore del BD espresso in pixel (0 - 199)
.byte   regAddr       ;indirizzo del registro (r5 - r10, r15)
```

DBGETSTRING - Dialog Box Get Text String (Input di una stringa con BD)

Codice del comando: 13

Riceve una stringa di testo proveniente in input dall'utente e la memorizza nel buffer indirizzato dal registro il cui indirizzo è specificato in regAddr. Il comando riporta anche l'eco sullo schermo, nella posizione indicata dalle coordinate xpos_offset e ypos_offset espresse in pixel e riferite all'angolo sinistro in alto del BD. maxChars è il massimo numero di caratteri memorizzabili nel buffer.

```
.byte   DBGETSTRING   ;byte comando
.byte   xpos_offset   ;offset dal lato sinistro del BD espresso in pixel (0 - 256)
.byte   ypos_offset   ;offset dal lato superiore del BD espresso in pixel (0 - 199)
.byte   regAddr       ;indirizzo del registro (r5 - r10, r15)
.byte   maxChars      ;massimo numero di caratteri da memorizzare nel buffer
```

DBSYSOPV - Dialog Box System Other Press Vector

Codice del comando: 14

Questo comando permette all'utente di chiudere il BD premendo il pulsante del mouse quando questo si trova in un'area diversa da un'icona. Se per esempio il BD è aperto semplicemente per visualizzare i nomi degli autori dell'applicazione, non c'è bisogno di alcuna icona e l'utente può chiudere il BD premendo il pulsante del mouse in un'area qualunque.

```
.byte   DBSYSOPV      ;byte comando
```

DBGRPHSTR - Dialog Box Graphics String

Codice del comando: 15

Questo comando ordina a GEOS di eseguire la stringa grafica (graphics string) indicata da grphStrPtr.

```
.byte   DBGRPHSTR     ;byte comando
.word   grphStrPtr    ;puntatore alla stringa grafica da eseguire
```

DBGETFILES - Dialog Box Get Files (Selezione dei file nei BD)

Codice del comando: 16

Questo comando apre all'interno del BD una finestra che elenca nomi di file. Al suo interno GEOS elenca una serie di nomi di file letti da disco e raggruppati secondo una chiave di ricerca. L'utente può selezionare uno qualunque dei nomi di file visualizzati. Il nome di file selezionato viene copiato nel buffer indicato dal registro r5. La "chiave" per la ricerca dei file i cui nomi devono essere visualizzati nella finestra è il tipo del file. Normalmente si impiega questo sistema di selezione per permettere all'utente di scegliere un file fra tutti quelli dello stesso tipo presenti sul disco. Per esempio, quando l'utente vuole selezionare il driver di input in deskTop, all'interno del box di selezione compaiono esclusivamente i file di tipo INPUT_DEVICE. Il tipo di file che si desidera impiegare come chiave di ricerca dev'essere memorizzato in r7L. Esiste anche la possibilità di indicare una seconda chiave, utile per raggruppare file creati da una stessa applicazione. Con questa seconda chiave, subordinata alla prima, il campo di ricerca su disco

dei file si restringe ulteriormente. Di solito le applicazioni creano file di tipo APPL_DATA. Questa informazione non è però sufficiente per raggruppare solo i file di tipo APPL_DATA di una particolare applicazione. Fortunatamente a ogni file, oltre il tipo del file, è associato anche un nome permanente (permanent name) che ne specifica ulteriormente la provenienza. Questo nome permanente è la seconda chiave di ricerca. Memorizzando in r10, di solito azzerato, l'indirizzo di una stringa a terminazione nulla che contenga il nome permanente, si genera la seconda chiave di ricerca che restringe il campo di raggruppamento. In questo modo GEOS raggruppa tutti i file del tipo specificato e provenienti dalla stessa applicazione. Il nome permanente, come vedremo, è memorizzato nel File Header del file ed è generato dall'applicazione che ha creato il file. Se per esempio r10 indica il nome permanente "Paint Image", all'interno della finestra di selezione compariranno esclusivamente i file generati dall'applicazione geoPaint. Se sullo stesso disco vi sono più file di quanti ne possa visualizzare contemporaneamente la finestra, compaiono nella parte sottostante alla finestra due frecce indicanti le due direzioni verticali di scroll della lista all'interno della finestra. Questo comando è in grado di creare una lista contenente un numero massimo di 16 file.

```
.byte   DBGETFILES    ;byte comando
.byte   xpos_offset   ;offset dal lato sinistro del BD in pixel (0 - 255) della finestra
.byte   ypos_offset   ;offset dal lato superiore del BD in pixel (0 - 199) della finestra
r7L     - tipo del file in ambiente GEOS
r5      - puntatore al buffer al quale restituire il nome del file selezionato
r10     - puntatore al nome permanente (stringa a terminazione nulla) del file
          per restringere il campo di ricerca
```

DBOPVEC - Dialog Box User Other Press Vector

Codice del comando: 17

Questo comando permette all'applicazione di specificare in userVector l'indirizzo di una routine da eseguire se l'utente preme il pulsante del mouse in un'area del BD diversa da un'icona. Questa routine può terminare con un rts restituendo il controllo ai codici di gestione del BD, o può eseguire l'istruzione jmp RstrFrmDialogue per chiudere il BD e restituire il controllo all'applicazione.

```
.byte   DBOPVEC       ;byte comando
.word   userVector    ;otherPressVec definito dall'utente
```

DBUSRICON - Dialog Box User Icon (Icona non standard per i BD)

Codice del comando: 18

Questo comando permette all'applicazione di definire e visualizzare all'interno del BD un'icona diversa da quelle standard per i BD. La tavola di definizione dell'icona (Usr_Icn_Tab) include i puntatori ai dati grafici dell'icona e alla routine di servizio associata. Quando l'icona viene attivata, GEOS effettua un jsr alla routine di servizio associata.

Questa routine può eseguire un jmp RstrFrmDialogue per chiudere il BD o effettuare un rts per ricedere il controllo ai codici di gestione del BD. In quest'ultimo caso l'applicazione deve avere previsto all'interno del BD qualche altro modo per chiuderlo, per esempio una differente icona. La routine, detenendo il controllo, può anche riconfigurare interamente il BD.

```
.byte   DBUSRICON     ;byte comando
.byte   xpos_offset   ;coordinata x dell'icona in byte riferita al lato sinistro
                      ;del BD (0 - 39)
.byte   ypos_offset   ;coordinata y dell'icona in pixel riferita al lato superiore
                      ;del BD (0 - 199)
.word   Usr_Icn_Tab   ;puntatore alla tavola di definizione dell'icona

Usr_Icn_Tab:

.word   iconPic       ;puntatore ai dati grafici dell'icona
.byte   0             ;coordinata x (ignorata)
.byte   0             ;coordinata y (ignorata)
.byte   width_bytes   ;la larghezza di 6 byte e' quella di default per le icone dei BD
.byte   height_pixels ;l'altezza di 16 pixel e' quella di default per le icone dei BD
.word   Serv_Routine  ;routine di servizio associata all'icona
```

DB USR ROUT - Dialog Box User Routine

Codice del comando: 19

Questa funzione permette di personalizzare completamente un BD. La routine puntata da userVector viene eseguita appena dopo che il BD è stato visualizzato, e può effettuare qualunque tipo di operazione riguardi il BD. Per restituire il controllo ai codici di gestione dei BD, può effettuare un rts, a meno che non sia stato predisposto un altro modo per chiudere il BD e l'utente l'abbia selezionato. Si tratta comunque di un comando che offre molte possibilità diverse, e torneremo a parlarne in seguito.

```
.byte   DB_USR_ROUT   ;byte comando
.word   userVector    ;puntatore alla routine definita dall'applicazione di gestione del BD
```

DBOPVEC

DBOPVEC imposta un vettore che contiene l'indirizzo di una routine da eseguire quando l'utente preme il pulsante del mouse in un'area diversa da un'icona. La routine può terminare con un rts, nel qual caso il controllo viene restituito ai codici di gestione dei BD contenuti in MainLoop. In seguito si possono eseguire altri comandi per le icone, per i box di dialogo o selezioni di icone.

Se il programmatore desidera che la routine termini chiudendo il BD e restituendo il controllo all'applicazione, come accade con il comando DBSYSOPV, deve terminare la routine con l'istruzione jmp RstrFrmDialogue. Quando la routine termina restituendo il controllo all'applicazione, deve memorizzare nella variabile globale sysDBData un codice di riferimento per l'applicazione che chiarisca la ragione per cui il BD è stato chiuso. RstrFrmDialogue trasferisce il contenuto di questa variabile globale in r0L. Nel caso che l'utente possa chiudere il BD in modi diversi, è importante che l'applicazione riceva in r0L il codice del particolare comando impartito. La routine prevista dall'applicazione per servire il comando DBOPVEC, deve restituire in sysDBData il valore del codice di riferimento assunto dall'applicazione per descrivere la causa della chiusura del BD. Questo codice non dev'essere uno di quelli utilizzati quando l'utente seleziona una delle icone standard per i BD. Dal momento che un BD non può gestire più di otto icone contemporaneamente, è sufficiente associare codici maggiori di 8 alle altre condizioni di uscita previste dalla routine di servizio del comando.

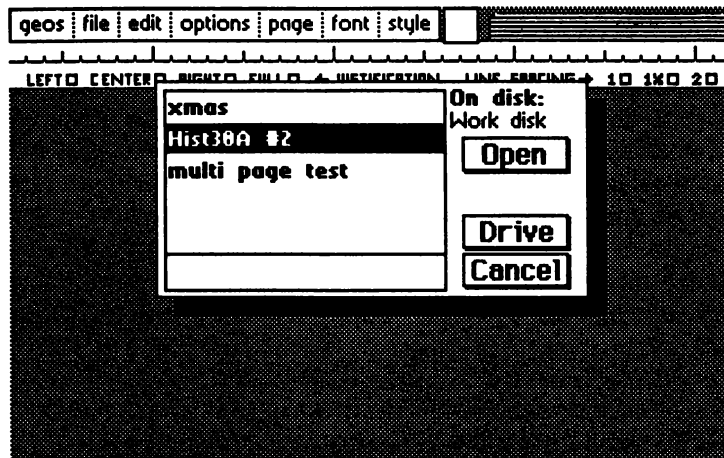
DBUSRICON

Se l'applicazione deve definire all'interno del BD un'icona diversa da quelle standard, può utilizzare il comando DBUSRICON per definirla e attivarla. Nei vari parametri che seguono il byte comando vi è una word che specifica l'indirizzo della tavola di definizione dell'icona. La struttura di questa tavola di definizione presenta sostanziali differenze da quella utilizzata per definire le icone di un'applicazione. Come si può osservare nella precedente descrizione del comando, i byte coordinate x e y all'interno della tavola di definizione sono impostati a zero. GEOS per collocare l'icona all'interno del BD utilizza i byte posizione specificati appena dopo il byte comando. La routine di servizio associata all'icona è indicata all'interno della tavola di definizione e viene eseguita non appena si seleziona l'icona. Come avviene per DBOPVEC, anziché restituire il controllo all'applicazione come fanno le icone di sistema, queste icone definite dall'applicazione restituiscono il controllo ai codici di gestione dei BD presenti in MainLoop, eseguendo l'istruzione rts. Il BD può contenere altre icone, per esempio di sistema, che se vengono selezionate chiudono il BD, come CANCEL o OK.

Se invece il programmatore desidera che l'icona definita dall'applicazione chiuda il BD e restituisca il controllo al codice che ha fatto la chiamata, deve concludere la

routine di servizio associata all'icona con l'istruzione jmp RstrFrmDialogue. Come accade per il comando DBOPVEC, la routine associata a DBUSRICON dovrebbe memorizzare nella variabile globale sysDBData il codice di chiusura che successivamente RstrFrmDialogue trasferisce in r0L. Questo numero dev'essere diverso da quelli assegnati alle icone di sistema, per non creare ambiguità.

DBGETFILES

Il comando DBGETFILES è sicuramente il più importante comando disponibile per i BD. La figura che segue illustra l'impiego di questo comando.

Possiamo qui osservare la finestra all'interno del BD che elenca i nomi di alcuni file. Se esistono sul disco più file di quanti la finestra ne possa contenere, appaiono nella parte bassa del BD due frecce con le punte una verso il basso e una verso l'alto. Agendo su di esse con il mouse, l'utente può scorrere l'intera lista di nomi, anche se la finestra ne visualizza solo una parte. In questo modo non possono essere raggruppati più di 16 nomi di file ma di solito è già una quantità sufficiente. Quando il BD è aperto e il comando viene eseguito, il registro r7L deve contenere il tipo di file secondo lo standard GEOS (SYSTEM, DESK_ACC, APPLICATION, APPL_DATA, FONT, PRINTER, INPUT_DEVICE, DISK_DEVICE...). Questa è la chiave principale di raggruppa-

mento dei nomi; r5 deve puntare a un buffer entro il quale sarà restituito il nome del file selezionato. Se l'applicazione, prima di attivare il BD, memorizza in questo buffer una stringa a terminazione nulla contenente il nome di un file, questo sarà il primo a essere visualizzato nel box, e come tale sarà scritto in negativo.

Selezionare un file significa memorizzarne il nome nel buffer puntato da r5 e farlo diventare una stringa a terminazione nulla (terminandolo con uno 0). Se r10 è 0 vengono raggruppati tutti i file del tipo specificato. Se invece r10 punta a una stringa a terminazione nulla, questa viene assunta come seconda chiave di ricerca e rappresenta il nome permanente (permanent name) associato al file. Tramite questa seconda chiave di selezione è possibile restringere la ricerca di file dello stesso tipo raggruppando esclusivamente quelli creati dalla stessa applicazione. La stringa "nome permanente" è memorizzata nel blocco File Header del file e dovrebbe essere generata dall'applicazione all'atto della creazione del file. Essa contiene un nome generato dall'applicazione e dovrebbe identificare tutti i file creati dalla stessa applicazione. Per esempio, a geoPaint dovrebbe interessare aprire e gestire solo file di sua creazione, e non tutti i possibili file dati presenti sul disco. Se geoPaint, all'apertura di un BD per selezionare il file su disco, specifica in r10 l'indirizzo della stringa "Paint Image", verranno raggruppati solo i file con quel nome permanente. Questa seconda chiave di ricerca, subordinata alla prima, diventa particolarmente utile quando si desidera raggruppare, per esempio, solo quei file di tipo APPL_DATA generati da una particolare applicazione.

Dal momento che un esempio è spesso molto più chiaro di una spiegazione, più avanti riportiamo una piccola tavola di definizione che illustra l'impiego di questo comando.

DB_USR_ROUT

Il comando DB_USR_ROUT ordina a GEOS di chiamare una routine di gestione del BD, elaborata dall'applicazione, quando il BD è stato visualizzato. La routine può essere anche molto complessa. Può per esempio impostare processi temporizzati, visualizzare finestre e così via. Dal momento che DoDlgBox e RstrFrmDialogue, rispettivamente, salvano e ripristinano lo stato del sistema, la routine di gestione del BD associata al comando DB_USR_ROUT non rischia di alterare in alcun modo lo stato della macchina precedente alla chiamata di DoDlgBox. In appendice sono evidenziate tutte le aree di memoria che vengono salvate durante l'apertura di un BD, e nella descrizione della routine LdDeskAcc (vedere il capitolo 15) è illustrata una lista di queste aree. Il motivo per cui queste aree vengono descritte nell'ambito della routine LdDeskAcc risiede nella stretta similitudine che lega i desk accessory ai box di dialogo: per entrambi, prima che vengano aperti, GEOS salva le aree di memoria che caratterizzano lo stato del sistema in una particolare zona della memoria, trasparente alle applicazioni. L'unica limitazione dei box di dialogo consiste nell'impossibilità di chiamare la

routine DoIcons se all'interno del BD sono state già attivate le icone standard, dal momento che i due set di icone interferirebbero l'uno con l'altro.

La struttura di icone possiamo attivarla e visualizzarla, dopo l'esecuzione del comando DB_USR_ROUT, grazie a comandi di altro tipo. Solo in questo modo un'icona può essere visualizzata sopra un disegno realizzato dalla routine associata al comando DB_USR_ROUT. La routine, se si desidera che gli altri comandi che la seguono nella tavola di definizione del BD siano eseguiti, deve restituire il controllo ai codici di gestione dei BD presenti in MainLoop tramite l'istruzione rts.

La chiusura di un box di dialogo

La parte di schermo coperta dal BD aperto può essere visualizzata di nuovo in due modi. Se lo schermo principale è stato copiato nel buffer di schermo, è sufficiente chiamare la routine RecoverRectangle che copia un rettangolo dal buffer di schermo allo schermo principale. Questa operazione viene automaticamente effettuata dal Kernel. Se invece il flag dispBufferOn è impostato in modo tale che il buffer di schermo contenga soltanto codici, e non una copia dello schermo principale, l'applicazione deve provvedere a un altro modo per ripristinare lo schermo precedente all'apertura del BD.

Quando viene eseguita, RstrFrmDialogue chiama la routine il cui indirizzo è memorizzato nel vettore recoverVector. Normalmente questo vettore contiene l'indirizzo della routine RecoverRectangle. Per ripristinare lo schermo principale dopo la chiusura di un BD, devono essere effettuate due chiamate attraverso recoverVector. La prima, effettuata da RstrFrmDialogue, è eseguita con le coordinate del box ombra del BD, ripristinando così l'area coperta dall'ombra del BD. La seconda chiamata attraverso recoverVector, sempre effettuata da RstrFrmDialogue, viene eseguita con le coordinate del BD. In questo modo si ripristina completamente l'area coperta dal BD e dalla sua ombra.

Se l'applicazione non utilizza il buffer di schermo per mantenere una copia dello schermo principale, deve memorizzare in recoverVector un ulteriore indirizzo di una routine che ripristini in qualche modo lo schermo principale. Le dimensioni dell'area da ricreare vengono passate negli stessi registri utilizzati da RecoverRectangle: r2 - r4. La routine di "ripristino schermo", puntata da recoverVector, sarà chiamata due volte e potrà utilizzare le dimensioni dell'area da ripristinare contenute nei registri r2 - r4. Se la routine risolve il problema con una sola esecuzione, deve contenere un flag che alla seconda chiamata le faccia effettuare un semplice rts.

Esempio di box di dialogo: OpenBox

Descrizione: Questa tavola di definizione è tratta da geoWrite e definisce un box di dialogo che permette all'utente di creare un nuovo documento, di aprire un documento già esistente o di terminare l'applicazione.

```
OpenBox:
 .byte    DEF_DB_POS|1        ;BD standard realizzato da GEOS e matrice grafica 1 per l'ombra

 .byte    DBTXTSTR            ;visualizza la stringa
 .byte    TXT_LN_X            ;posiziona la stringa all'offset orizzontale standard
                              ;(16 pixel)
 .byte    2*8                 ;offset y in pixel dal lato superiore dello schermo
 .word    selectOptionTxt     ;puntatore al messaggio "Please Select Option:"

 .byte    DBUSRICON           ;icona CREATE definita dal programmatore
 .byte    2                   ;offset x in byte dal lato sinistro del BD al lato sinistro
                              ;dell'icona
 .byte    3*8                 ;offset y in pixel dal lato superiore del BD al lato
                              ;superiore dell'icona
 .word    createIcon          ;puntatore alla tavola di definizione dell'icona CREATE
 .byte    DBTXTSTR            ;visualizza il testo "new document"
 .byte    (2+6)*8+7           ;offset x in pixel: (2 + larghezza icona) * 8 pixel + 7
 .byte    3*8+10              ;offset y in pixel: 10 pixel sotto il lato superiore
                              ;dell'icona CREATE
 .word    newOptionTxt        ;puntatore al testo "new document"

 .byte    OPEN                ;icona standard di sistema OPEN
 .byte    2                   ;offset x in byte
 .byte    6*8                 ;offset y in pixel: 24 pixel sotto l'icona CREATE

 .byte    DBTXTSTR            ;posiziona il testo "existing document" alla destra
                              ;dell'icona
 .byte    (2+6)*8+7           ;offset x in pixel: (2 + larghezza icona) * 8 pixel + 7
 .byte    6*8+10              ;offset y in pixel: 24 pixel al di sotto "new document"
 .word    openOptionTxt       ;puntatore "existing document"
```

```
 .byte    DBUSRICON          ;l'applicazione definisce un'icona personalizzata
 .byte    2                  ;offset x in byte per il lato sinistro dell'icona
 .byte    9*8                ;offset y in pixel per il lato superiore dell'icona
 .word    quitIcon           ;puntatore alla tavola di definizione dell'icona QUIT

 .byte    DBTXTSTR           ;visualizza il testo "to deskTop"
 .byte    (2+6)*8+7          ;offset x: (2 + larghezza dell'icona) * 8 pixel + 7
 .byte    9*8+10             ;offset y in pixel: 24 pixel sotto il lato superiore
                             ;dell'icona Open
 .word    quitOptionTxt      ;puntatore al testo "to deskTop"

 .byte    0                  ;fine della tavola di definizione del BD

selectOptionTxt:             ;la stringa contiene i caratteri di controllo dello stile
 .byte    PLAINTEXT, BOLDON, "Please Select Option:", PLAINTEXT, 0

newOptionTxt:                ;notare che ogni stringa e' a terminazione nulla
 .byte    "new document", 0

openOptionTxt:
 .byte    "existing document", 0

quitOptionTxt:
 .byte    "to deskTop", 0

createIcon:                  ;tavola di definizione dell'icona CREATE
 .word    createIconPic      ;indirizzo dei dati grafici del disegno dell'icona CREATE
 .byte    0                  ;coordinata x, qui ignorata, dell'icona
 .byte    0                  ;coordinata y, qui ignorata, dell'icona
 .byte    SYSDBI_WIDTH       ;utilizza la larghezza standard delle icone
                             ;di sistema (6 byte)
 .byte    SYSDBI_HEIGHT      ;utilizza l'altezza standard delle icone
                             ;di sistema (16 pixel)
 .word    createIconServ     ;puntatore alla routine di servizio associata che crea
                             ;il file e ritorna all'applicazione
quitIcon:                    ;tavola di definizione dell'icona QUIT
 .word    quitIconPic        ;indirizzo dei dati grafici del disegno dell'icona QUIT
 .byte    0                  ;coordinata x, qui ignorata, dell'icona
 .byte    0                  ;coordinata y, qui ignorata, dell'icona
 .byte    SYSDBI_WIDTH       ;utilizza la larghezza standard delle icone
                             ;di sistema (6 byte)
```

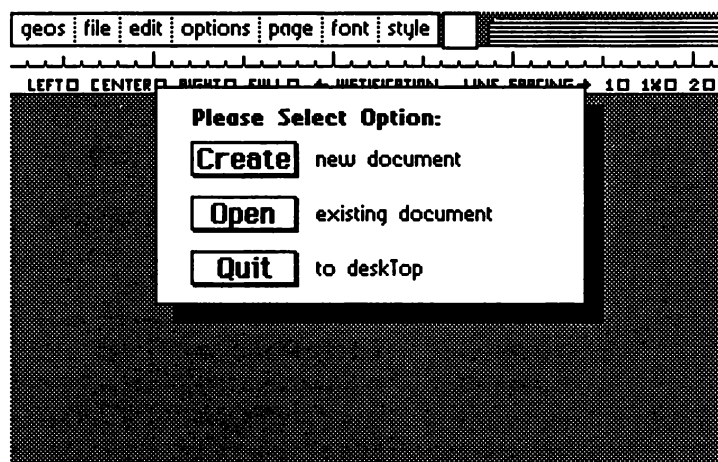
```
 .byte    SYSDBI_HEIGHT      ;utilizza l'altezza standard delle icone
                             ;di sistema (16 pixel)
 .word    quitIconServ       ;puntatore alla routine di servizio che termina
                             ;l'applicazione e cede il controllo a deskTop
createIconServ:              ;la semplice routine di servizio per l'icona CREATE
 lda      #1                 ;indica il numero dell'icona come se fosse stata attivata
                             ;l'icona OK
 bne      jmpIconServ

quitIconServ:
 lda      #2                 ;restituisce il valore per l'icona QUIT

jmpIconServ:
 sta      sysDBData          ;memorizza il numero dell'icona prima
                             ;di chiamare RstrFrmDialogue
 jmp      RstrFrmDialogue    ;chiusura del BD

createIconPic:
 96 byte di dati grafici per l'icona che non riportiamo

quitIconPic:
 96 byte di dati grafici per l'icona che non riportiamo
```

Ecco come si presenta il BD che abbiamo appena definito:

L'esempio che segue mostra un BD per la selezione di file da disco.

Esempio di box di dialogo: LoadBox

Descrizione: Questo esempio, tratto da geoWrite, è una tavola di definizione di un BD visualizzato per aprire un documento già esistente. Se dal BD dell'esempio precedente l'utente seleziona l'icona OPEN, l'applicazione visualizza il BD per la scelta del file da aprire.

Note:

r7L	contiene il tipo del file, secondo lo standard GEOS, che rappresenta la chiave di selezione dei file su disco
r5	punta al buffer dove viene memorizzato il nome del file selezionato dall'utente
r10	puntatore alla stringa che contiene il nome permanente associato al file. Questo nome rappresenta la seconda chiave di selezione dei file su disco. In questo caso "geoWrite". Se il puntatore è azzerato il nome permanente non viene preso in considerazione.

```
LoadBox:
  .byte    DEF_DB_POS|1        ;BD di dimensioni standard, con l'ombra visualizzata
                               ;impiegando la matrice grafica 1. Il box di selezione dei file
                               ;(GetFile Box) si adatta molto bene a essere utilizzato
                               ;con i BD di dimensioni standard
  .byte    DBTXTSTR            ;visualizza la stringa "On disk:"
  .byte    DBI_X_2 * 8 - 6     ;offset x in pixel per il testo (= 17*8-6 = 130)
  .byte    TXT_LN_1_Y - 6      ;offset y
  .word    curDiskString       ;puntatore al testo "current Disk:"

  .byte    DBTXTSTR            ;stringa che contiene il nome del disco corrente
  .byte    DBI_X_2 * 8 - 6     ;offset x in pixel per il testo (= 17*8-6 = 130)
  .byte    TXT_LN_2_Y - 12     ;offset y
  .word    diskName            ;puntatore al buffer che gia' deve contenere il nome del disco

  .byte    DBGETFILES          ;il comando di selezione di file su disco
  .byte    4                   ;offset x in pixel dal lato sinistro del BD
  .byte    4                   ;offset y in pixel dal lato superiore del BD
```

```
 .byte    OPEN       ;visualizza l'icona OPEN
 .byte    DBI_X_2    ;offset x standard per l'icona OPEN
 .byte    24         ;offset y dal lato superiore del BD

 .byte    CANCEL     ;visualizza l'icona standard per l'icona CANCEL
 .byte    DBI_X_2    ;offset x standard per l'icona CANCEL
 .byte    72         ;offset y

 .byte    DISK       ;icona DISK, quando sono connessi due drive la routine
                     ;di servizio accede all'altro disk drive
 .byte    DBI_X_2    ;offset x standard per l'icona DISK
 .byte    48         ;offset y per l'icona

 .byte    0          ;termine della tavola di definizione del BD

curDiskString:
 .byte    PLAINTEXT, BOLDON, "On disk:", PLAINTEXT, 0   ;la stringa da visualizzare nel BD
```

Le routine di gestione dei box di dialogo

Nelle pagine successive sono riportate le descrizioni complete di tutte le routine di gestione dei BD che GEOS mette a disposizione delle applicazioni.

DoDlgBox

Funzione: Visualizza un box di dialogo.

Indirizzo: $C256

Parametri: r0L restituito con il valore che indica quale selezione è stata operata dall'utente per chiudere il BD. Se l'utente ha selezionato un'icona di sistema, r0L riporta il numero che la identifica. DoDlgBox di fatto non esegue alcun ritorno all'applicazione. Essa salva lo stato del sistema, in modo che MainLoop possa dedicarsi temporaneamente al nuovo ambiente che si viene a creare con l'apertura del BD, visualizza il BD e cede il controllo a MainLoop che gestisce i nuovi eventi. Fra i vari eventi definiti per il BD, quelli che determinano condizioni di uscita utilizzano la routine RstrFrmDialogue per cedere il controllo all'applicazione appena dopo la chiamata a DoDlgBox. Quindi i codici che chiamano DoDlgBox ottengono di nuovo il controllo solo dopo che il BD è stato richiuso. La chiamata viene effettuata con l'istruzione jsr DoDlgBox

Restituisce: La routine non restituisce direttamente alcun parametro perché il ritorno, comandato dalla selezione dell'utente, passa attraverso la routine RstrFrmDialogue

Distrugge: a, x, y, r0 - r4, lo stack è preservato come tutte le variabili globali di sistema

Sinossi: DoDlgBox visualizza un BD. Il codice che effettua la chiamata deve passare in r0 l'indirizzo della tavola di definizione del BD. Tutte le variabili di sistema, cioè l'intero stato del sistema (disk, menu, icone, processi...) eccetto i registri r0 - r4, vengono salvate per essere successivamente ripristinate da RstrFrmDialogue. Le routine realizzate per gestire il BD possono creare un nuovo ambiente di lavoro che comprende processi, menu e icone, senza rischiare di alterare lo stato del sistema precedente all'apertura del BD. I registri r5 - r10 possono essere utilizzati dall'applicazione per comunicare informazioni al BD prima che sia attivato. Nell'appendice B, tutte le locazioni di memoria racchiuse fra le label s_zp_global e e_zp_global indicano le aree di memoria che vengono salvate durante l'attivazione dei BD e degli accessori da scrivania (Desk Accessory).

RstrFrmDialogue

Funzione: Effettua la chiusura di un BD ed il ritorno ai codici dell'applicazione.

Indirizzo: $C2BF

Chiamata da: I BD possono essere chiusi fondamentalmente in due modi. Se l'utente seleziona un'icona di sistema o attiva un comando di chiusura del BD, viene automaticamente eseguita RstrFrmDialogue. Oltre alla chiusura standard prevista dal sistema, il BD può essere disattivato anche attraverso routine realizzate per gestirlo in maniera non convenzionale. In questi casi i codici dell'applicazione devono chiamare la routine RstrFrmDialogue per chiudere definitivamente il BD e ripristinare lo stato precedente del sistema. Quindi, a prescindere dal fatto che la disattivazione sia standard o realizzata dall'applicazione, la chiusura del BD avviene sempre attraverso la routine RstrFrmDialogue

Parametri: sysDBData questa variabile di sistema deve contenere l'identificatore della selezione che ha causato la chiusura del BD. Se la disattivazione del BD è avvenuta attraverso la selezione di un'icona, sysDBData deve contenere il numero di questa icona in modo che RstrFrmDialogue lo restituisca in r0L. Questa informazione consente al codice che ha richiesto l'apertura del BD, d'identificare la selezione operata dall'utente

Restituisce: r0L questo registro viene aggiornato con il contenuto della variabile globale sysDBData. Normalmente restituisce l'identificatore della selezione che ha causato la chiusura del BD

Altro dal momento che i BD possono essere gestiti anche da routine dell'applicazione, può accadere che esse restituiscano parametri specifici a seconda delle operazioni che hanno compiuto

System all'apertura del BD, DoDlgBox salva l'intero stato del sistema, quindi RstrFrmDialogue lo deve ripristinare per ricreare la situazione precedente all'apertura del BD. Perciò, a eccezione degli pseudoregistri, lo stato del sistema memorizzato su RAM (menu, icone, buffer, ecc) viene integralmente ripristinato

Distrugge: a, x, y, r0H - r14

Sinossi: Questa routine viene chiamata per chiudere il BD. Può essere eseguita sia dai codici di gestione del BD presenti nel Kernel di GEOS, sia dalle routine (se presenti) previste dall'applicazione per operare una gestione non standard del BD. Per il codice che ha richiesto l'apertura del BD, la chiamata di questa routine è completamente trasparente, nel senso che il codice esegue l'istruzione jsr DoDlgBox e ottiene nuovamente il controllo all'istruzione seguente, una volta che il BD è stato chiuso. Per esempio, il codice dell'applicazione richiede l'apertura del BD. Allora GEOS salva lo stato del sistema e tratta il BD come un nuovo ambiente di lavoro entro il quale, come avviene di solito, una tavola di definizione indica tutti gli eventi che il Kernel deve gestire (menu, icone, processi...). Alcuni eventi sono standard e non richiedono alcuna routine di servizio da parte dell'applicazione, per essere eseguiti; una volta selezionati gli eventi, le routine di sistema che li gestiscono memorizzano in sysDBData l'identificatore dell'evento standard attivato e chiamano RstrFrmDialogue per chiudere il BD e ripristinare lo stato del sistema come se niente fosse accaduto. Altri eventi invece possono essere realizzati dall'applicazione per soddisfare particolari esigenze, e non devono necessariamente chiudere il BD. A essi sono associate le appropriate routine di servizio messe a disposizione dall'applicazione. Le routine di servizio che, se attivate, chiudono il BD devono aggiornare sysDBData con l'identificatore dell'evento e devono contenere la chiamata alla routine RstrFrmDialogue. Quando viene eseguita RstrFrmDialogue, lo stato del sistema viene interamente ripristinato, a eccezione degli pseudoregistri, e in r0L è riportato l'identificatore dell'evento che ha causato la chiusura del BD.

13 IL SISTEMA DI GESTIONE DEI FILE

GEOS adotta un sistema di organizzazione dei file su disco che ha molto in comune con il DOS residente nel 1541, ma aggiunge alla struttura di base alcune caratteristiche proprie. Vediamo i fattori grazie ai quali abbiamo potuto rendere più efficiente la gestione dei file e l'accesso al disco.

La struttura originale del C-64 non era stata realizzata per l'impiego di una memoria di massa su disco. In origine, non si pensava affatto che il disk drive del C-64 avrebbe avuto una così grande diffusione. La conseguente lentezza di accesso al disco si è dimostrata col tempo uno dei limiti più evidenti della macchina. A causa di questa lentezza, le applicazioni create per il C-64 di solito leggono i dati dal disco all'inizio dell'esecuzione e durante l'elaborazione li salvano il meno possibile, rimandando l'aggiornamento dei file a un secondo momento. Se durante l'esecuzione dell'applicazione si rende necessario accedere al disco, l'utente può tranquillamente concedersi un caffè, tanta è la lentezza che caratterizza l'accesso. Per ovviare a questo problema, GEOS mette a disposizione un codice turbo (diskTurbo) che accresce notevolmente la velocità durante lo scambio dei dati fra il C-64 e il disk drive, diminuendo di conseguenza i tempi di accesso. Con l'introduzione di questa routine, l'accesso al disco diventa più pratico e più veloce. Lo scambio di dati può interessare anche solo parti di file e può avvenire tranquillamente durante l'esecuzione delle applicazioni in quanto, in ambiente GEOS, il tempo necessario per comunicare con il disco è minimo.

GEOS mette a disposizione due strutture fondamentali per organizzare i file su disco. La prima è abbastanza simile alla struttura convenzionale dei file adottata dal C-64 e viene chiamata struttura SEQUENTIAL(†). Essa prevede che i file siano organizzati e

---

(†) Il termine SEQUENTIAL, utilizzato per identificare una delle due strutture possibili previste da GEOS, identifica qualsiasi tipo di file GEOS con una struttura non VLIR, e non dev'essere confuso con il tipo SEQ disponibile nel DOS del 1541. Per fare un esempio, i tipi USR, PRG e SEQ utilizzati dal C-64 rientrano tutti nella struttura SEQUENTIAL.

gestiti secondo una lista di blocchi allocati su disco, che raccolgono sequenzialmente i dati del file. Prima di proseguire è bene precisare il significato della parola "blocco". Un blocco per il C-64, a prescindere dalla sua collocazione (memoria interna o disco), rappresenta un gruppo di dati significativi composto da 256 byte, numerati dal byte 0 al byte 255 ($FF).

Dal momento che blocco e settore, su disco, hanno la stessa dimensione, nel corso del libro adottiamo indifferentemente entrambi i termini. Volendo essere precisi, il "settore" è uno spazio fisico sul disco, mentre il "blocco" è un gruppo di dati che viene memorizzato nel settore, e lo occupa completamente. Utilizzeremo comunque i due termini, senza distinzione, per indicare un gruppo di dati da 256 byte memorizzato su disco. La parola blocco, dal momento che individua un gruppo di dati, si adatta perfettamente a rappresentare anche i dati presenti in memoria, che all'occorrenza possono essere suddivisi in gruppi di 256 byte (pagine). In quest'ultimo caso il termine settore sarebbe improprio, e quindi adotteremo unicamente la parola blocco. Ritornando alla struttura SEQUENTIAL, i primi due byte di ogni settore contengono l'indirizzo su disco del successivo settore concatenato. Questo indirizzo è indicato dal "numero di traccia" nel primo byte, e dal "numero di settore", nel secondo byte; quando ci riferiremo a questi due byte, li identificheremo con il termine "indirizzo T/S". L'ultimo blocco di un file su disco non segue questa regola. Dal momento che non ci sono blocchi successivi, e l'ultimo potrebbe non essere completamente riempito, si rende necessario indicare quanti byte dell'ultimo blocco sono da considerare significativi per il file di cui fa parte. Il primo byte, in questo caso, ha valore 0 (e questo identifica l'ultimo settore), e il secondo contiene un indice che punta all'ultimo byte significativo del file.

La seconda struttura di organizzazione dei file su disco è una novità per gli utenti del C-64. È chiamata struttura a Record Indicizzati di Lunghezza Variabile (Variable Lenght Indexed Record), che per brevità si riduce a "struttura VLIR". Per l'importanza che riveste, abbiamo dedicato un intero capitolo a questa nuova struttura da noi ideata e realizzata. In entrambe le strutture, GEOS alloca per ogni file un blocco addizionale che diventa parte integrante del file. Si tratta del blocco File Header, e contiene importanti informazioni che GEOS mette a disposizione di tutti i file, come per esempio i dati grafici dell'icona che individua il file sullo schermo e altri parametri che analizzeremo nelle prossime pagine.

Per riuscire a capire in modo davvero completo il sistema adottato da GEOS per i file, è necessario avere un'ottima conoscenza della struttura su cui si basa il sistema, cioè quella realizzata dalla Commodore. Per i lettori che non avessero mai approfondito questo argomento sono presenti in commercio diverse pubblicazioni dedicate al sistema dei file adottato per il C-64.

Questo capitolo è diviso in due sezioni principali. La prima è un veloce ripasso delle nozioni necessarie per intraprendere una lettura agevole del resto del capitolo. La seconda illustra nei dettagli il sistema dei file in ambiente GEOS e introduce le varie routine presenti nel Kernel per operare con il drive. Queste routine si possono

raggruppare in tre livelli, ciascuno contraddistinto da un crescente grado di completezza. Ogni livello è poi dettagliatamente illustrato nei capitoli successivi, un capitolo per ogni livello. Al livello più alto vengono descritte le routine più complete dal punto di vista operativo, in grado, di compiere autonomamente una sequenza di operazioni correlate. Sono le prime routine con cui il programmatore deve familiarizzarsi. Soddisfano le esigenze più comuni con una sola chiamata da parte dell'applicazione, ma essendo estremamente articolate è improbabile che possano soddisfare le richieste che escono dalla norma. A questo scopo esistono due livelli inferiori di routine che compiono operazioni sempre più semplici, ma che opportunamente combinate possono generare nuove funzioni di notevole complessità. Il compito del programmatore diventa più complesso, in questo caso, ma la flessibilità offerta dalle diverse combinazioni dovrebbe essere sufficiente a soddisfare qualunque esigenza. Dopo questi tre capitoli dedicati ai tre livelli di routine disponibili su GEOS, segue il capitolo riservato all'analisi della struttura VLIR di GEOS.

Le nozioni di base

Un disco in formato C-64 è suddiviso in 35 tracce. Le tracce sono fisicamente disposte in cerchi concentrici, e ogni anello circolare contiene più tracce. La traccia 1 si trova sul bordo del disco, la traccia 35 al centro. Ogni traccia è ulteriormente suddivisa in settori, chiamati anche blocchi. Le tracce situate lungo il bordo esterno del disco sono di maggiori dimensioni, rispetto a quelle vicine al centro, e quindi contengono un numero maggiore di settori. La tavola che segue elenca i settori per ogni gruppo di tracce e il numero di settori in esse contenuti.

Distribuzione dei settori nelle tracce

Numero di traccia	Numerazione dei settori	Numero di settori per traccia
da 1 a 17	da 0 a 20	21
da 18 a 24	da 0 a 18	19
da 25 a 30	da 0 a 17	18
da 31 a 35	da 0 a 16	17

La traccia 18 contiene la directory del disco. In essa sono presenti le informazioni associate ai file presenti su disco. Il settore 0 di questa traccia (blocco d'intestazione della directory, o Directory Header Block) contiene la mappa dei blocchi liberi (BAM) e l'intestazione della directory (Directory Header). La BAM è un insieme di bit, a ognuno dei quali è associato un settore. I bit corrispondenti a settori del disco allocati per contenere dati significativi sono impostati a 1, mentre quelli associati a settori liberi sono impostati a 0. Come per i file normali, anche la directory è una

concatenazione di settori, e quindi il Directory Header Block contiene l'indirizzo T/S del primo blocco dati della directory (Directory Block). L'organizzazione della BAM non viene assolutamente alterata da GEOS.

La Directory Header è un insieme di dati presente all'interno del Directory Header Block che contiene il nome del disco (una word chiamata Disk ID che costituisce l'identificatore del disco) e tre nuovi elementi introdotti da GEOS: un identificatore GEOS (GEOS ID) contenuto in una stringa di caratteri, l'indirizzo T/S del blocco addizionale associato al disco (Off Page Directory Block) che chiameremo per brevità Off Page del disco, e un byte che identifica il livello di protezione del disco. Queste tre informazioni introdotte da GEOS occupano uno spazio, nel Directory Header Block della directory, che normalmente non viene utilizzato.

La presenza della stringa GEOS ID indica che il disco è in formato GEOS, e contiene la versione del formato, che può essere importante per la compatibilità fra l'attuale versione di GEOS e quelle future. Questa stringa non dev'essere confusa con la GEOS ID memorizzata nel Kernel a $C000, che identifica non il formato delle strutture dei dati su disco ma la versione del Kernel di GEOS, come si è visto nel capitolo 1.

Il blocco Off Page associato al disco è un ampliamento al sistema dei file introdotto da GEOS e ha la stessa struttura dei blocchi che contengono la directory. Un blocco della directory contiene otto File Entry. Ogni file presente sul disco è individuato da un File Entry che gli assegna una serie di informazioni. In deskTop, quando si sposta l'icona di un file dal notes bianco alla scrivania (il bordo circostante), il File Entry del file viene cancellato dal blocco della directory che lo conteneva e trasferito nel blocco Off Page, che rappresenta la scrivania. Come ogni blocco della directory, Off Page non può contenere più di otto file. In memoria GEOS mantiene anche un buffer che all'occorrenza può raccogliere le informazioni relative a tutti i file presenti sulla scrivania. La scrivania, e quindi il blocco Off Page che la realizza su disco, è stata ideata e creata per consentire la copia dei file fra due dischi nel caso che il sistema sia dotato di un solo drive. Quando deskTop visualizza la directory a pagine di un nuovo disco, le icone che in precedenza erano state situate sulla scrivania non vengono cancellate dallo schermo. Se l'utente sposta una di queste icone in una pagina della nuova directory, deskTop inizia la procedura di copia del file dal disco precedente al disco nuovo, segnalando all'utente le opportune operazioni da compiere.

Il byte che identifica il livello di protezione del disco è memorizzato alla posizione 189 del Directory Header Block. Come sempre, nell'appendice si troverà la costante associata a questo byte, OFF_G_DTYPE = 189. Ricordiamo che la numerazione dei byte all'interno di un blocco inizia da 0. Di solito il byte di protezione è impostato a 0, per indicare che il disco non è soggetto ad alcun livello di protezione. Se invece contiene il codice ASCII della lettera P, il disco è soggetto al livello di protezione associato ai Master Disk. GEOS V1.3 e le applicazioni deskTop successive non permettono l'esecuzione di molte operazioni sui dischi Master Disk. Non possono essere formattati o copiati, per esempio, e i file in essi presenti non possono essere cancellati dal disco con la procedura normale (icona del file spostata sul cestino). I file possono comunque

essere spostati sulla scrivania e cancellati da lì. Con questo livello di protezione gli utenti del sistema operativo GEOS non rischiano di formattare accidentalmente un Master Disk, o di distruggere i dati che questo contiene. Se il byte individuato da OFF_GS_DTYPE contiene invece il codice ASCII della lettera B, il disco è un Boot Disk, cioè un disco sul quale risiede l'intero Kernel e i programmi per attivarlo.

Ecco la tavola che illustra nei dettagli il formato del Directory Header Block. Contiene la BAM e la Directory Header.

Formato della BAM e della Directory Header

Byte	Contenuto	Descrizione
0	18	Traccia del primo blocco della directory, e' sempre 18
1	1	Settore del primo blocco della directory, e' sempre 1
2	65	Codice ASCII di A indicante il formato del disco 1541
3		Ignorato
BAM Traccia 1		
4	21	Numero di settori in traccia 1
5		Bit dei settori 0-7
6		Bit dei settori 8-16
7		Bit dei settori 17-20
BAM Traccia 2		
8	21	Numero di settori in traccia 2
9		Bit dei settori 0-7
10		Bit dei settori 8-16
11		Bit dei settori 17-20
	... BAM per le tracce dalla 3 alla 34 ...	
BAM Traccia 35		
140	17	Numero di settori in traccia 35
141		Bit dei settori 0-7
142		Bit dei settori 8-16
143		Bit non usati per questa traccia

SEGUE

SEGUE

Directory Header

144 - 159		Nome del disco, completato con il carattere ASCII $A0
160 - 161	$A0	Spazi + tasto SHIFT (ASCII $A0)
162 - 163		Word Disk ID
164	$A0	Spazio + tasto SHIFT (ASCII $A0)
165 - 166	$32, $41	Codici ASCII 2A, Versione Dos e Formato Disco
167 - 170	$A0	Spazi + tasto SHIFT (ASCII $A0)
171 - 172		Indirizzo T/S del blocco Off Page della directory
173 - 188		Stringa GEOS ID: "GEOS format V1.2"
189	0/'P'/'B'	P indica il livello protezione Master Disk, B indica i Boot Disk
190 - 255	0	Attualmente inutilizzati

La tavola che segue mostra il formato di un blocco della directory (Directory Block), diverso dal blocco di intestazione. La struttura di base di un Directory Block resta inalterata in ambiente GEOS.

Struttura di un Directory Block

I blocchi della directory possono apparire in traccia 18 nei settori 1 - 19

Numero byte	Descrizione
0 - 1	Indirizzo T/S del successivo Directory Block
2 - 31	File Entry 1
32 - 33	Inutilizzati
34 - 63	File Entry 2
64 - 65	Inutilizzati
66 - 95	File Entry 3
96 - 97	Inutilizzati
98 - 127	File Entry 4
128 - 129	Inutilizzati
130 - 159	File Entry 5

SEGUE

SEGUE

```
160 - 161      Inutilizzati
162 - 191      File Entry 6
192 - 193      Inutilizzati
194 - 223      File Entry 7
224 - 225      Inutilizzati
226 - 255      File Entry 8
```

Nel formato originale del C-64 i File Entry contengono molti byte inutilizzati. GEOS li impiega per aumentare la quantità delle informazioni associate al singolo file. Analizziamo la struttura di un File Entry. Il byte 0, relativo all'inizio del File Entry indica il tipo del file secondo lo standard C-64. Di solito, i file in ambiente GEOS appartengono al tipo USR del C-64. Se la struttura del file non è VLIR, i byte 1 e 2 contengono l'indirizzo T/S del primo blocco di dati. Se invece la sua struttura è VLIR i byte 1 e 2 contengono l'indirizzo T/S del blocco indice del file (Index Table). I byte dal 3 al 18 contengono il nome del file. Se il nome contiene meno di 16 caratteri, i byte che restano devono contenere il carattere ASCII $A0. I byte 19 e 20 contengono l'indirizzo T/S del nuovo blocco che GEOS associa a ogni file: il blocco File Header. L'importanza di questo blocco aggiuntivo sarà illustrata più avanti. I byte 21 e 22 identificano il tipo di struttura e il tipo del file secondo il formato GEOS. Il byte 21 indica con quale struttura è organizzato il file su disco. Con il valore 0 si indica la struttura SEQUENTIAL e con il valore 1 la struttura VLIR. Il tipo del file indica a quali usi è destinato. DATA, BASIC, APPLICATION sono solo alcuni dei possibili tipi. Il tipo SYSTEM_BOOT dovrebbe essere assegnato solo ai file GEOS Boot e Kernel. Il tipo TEMPORARY viene assegnato ai file temporanei (swap file). Tutti i file di tipo TEMPORARY presenti su qualsiasi disco sono automaticamente cancellati da deskTop quando vi accede per leggere la directory. deskTop assume che siano presenti su disco per motivi accidentali. I file temporanei sono creati dalle applicazioni per usi interni e non hanno ragione di esistere se l'applicazione ha terminato il suo svolgimento. Non sempre però l'applicazione riesce a effettuare la cancellazione, per esempio quando viene a mancare improvvisamente l'energia elettrica, e in questi casi è il sistema che deve provvedere. Quando le applicazioni devono creare alcuni file per uso interno, devono utilizzare il tipo TEMPORARY e iniziare il nome del file con il carattere di controllo PLAINTEXT. Ecco un esempio:

```
swapName: .byte PLAINTEXT, "File Tempora.",0
```

In questo modo il nome del file viene visualizzato in deskTop con lo stile PLAINTEXT

e diventa impossibile che un altro file con lo stesso nome sia sostituito dal file temporaneo. I file di tipo AUTO_EXEC vengono automaticamente mandati in esecuzione dal Kernel di GEOS V1.3, e successivi, durante la fase d'inizializzazione del sistema. Il Kernel, prima di procedere al caricamento di deskTop, scorre la directory alla ricerca del primo file di tipo AUTO_EXEC, e se lo trova lo manda in esecuzione. Questo può effettuare variazioni nel sistema di default (configurazione, colori, espansioni...), e deve terminare con l'istruzione jmp EnterDeskTop. Infine troviamo i byte dal 23 al 27 che contengono l'ora e la data dell'ultima modifica al file. Così i file possiedono anche l'indicazione cronologica dell'ultima modifica.

Struttura del File Entry

Byte	Descrizione
0	Tipo file C-64: 0=DELeted, 1=SEQuential, 2=PRoGram, 3=USeR, 4=RELative. Il bit 6 indica se il file e' protetto
1 - 2	T/S del primo blocco di dati del file. Se il file e' a struttura VLIR, l'indirizzo T/S indica la Index Table
3 - 18	16 caratteri per il nome, completato eventualmente con il carattere ottenuto tramite il tasto SHIFT + Spazio (ASCII $A0)
19 - 20	T/S del blocco File Header (nuova struttura)
21	Struttura GEOS del file: 0=SEQUENTIAL, 1=VLIR
22	Tipo GEOS del file (vedere la tavola successiva)
23	Data: anno dell'ultima modifica; solo le ultime due cifre
24	Data: mese dell'ultima modifica (1 - 12)
25	Data: giorno dell'ultima modifica (1 - 31)
26	Data: ora dell'ultima modifica (0 - 23)
27	Data: minuto dell'ultima modifica (0 - 59)
28 - 29	Word che contiene il numero di settori occupati dal file

Tipi di file disponibili con GEOS

Codice	Nome associato al tipo
0	NOT_GEOS
1	BASIC
2	ASSEMBLY
3	DATA
4	SYSTEM
5	DESK_ACC
6	APPLICATION
7	APPL_DATA
8	FONT
9	PRINTER
10	INPUT_DEVICE
11	DISK_DEVICE
12	SYSTEM_BOOT
13	TEMPORARY
14	AUTO_EXEC (compatibile solo con versioni di GEOS superiori alla 1.2)

Il blocco File Header

Il blocco addizionale File Header che GEOS assegna a tutti i file rappresenta una novità per gli utenti Commodore. È stato creato per contenere i dati grafici che definiscono il disegno dell'icona e altre informazioni utili per GEOS e per l'utente. La sua ubicazione sul disco viene individuata dall'indirizzo T/S memorizzato nei byte 19 e 20 del File Entry del file. Il DOS del drive 1541 memorizza in questi due byte di un File Entry l'indirizzo T/S del side sector di gestione dei file di tipo RELative. Dal momento che GEOS non riconosce questo tipo di struttura, i byte 19 e 20 possono essere impiegati per puntare il blocco File Header associato al file.

I primi due byte di un settore di solito individuano l'indirizzo T/S del successivo settore concatenato, oppure, nel caso dell'ultimo settore, contengono il numero di byte significativi del settore stesso. Dal momento che il blocco File Header non è concatenato a nessun altro file, il byte 0 contiene il valore 0 e il byte 1 il valore $FF. In questo modo si indica che il blocco File Header costituisce un blocco extra del file e non è parte di alcuna concatenazione.

La tavola riportata nella pagina successiva analizza uno per uno i byte presenti nel blocco File Header.

Struttura del blocco File Header

Byte	Contenuto	Descrizione
0 - 1	$00, $FF	Il blocco File Header non e' concatenato a nessun altro blocco
2	3	Larghezza dell'icona in byte, sempre 3
3	21	Altezza dell'icona in linee di scansione, sempre 21
4	$80 + 63	Tipo dei dati in bit-map per il disegno dell'icona. Se il bit 7 e' impostato a 1, il valore contenuto nei restanti 7 bit indica il numero di singoli byte che seguono: sempre 63
5 - 67	$FF, $FF, $FF	Inizio dei 63 byte che definiscono il disegno dell'icona
	...	
	$FF, $FF, $FF	Fine dei dati che definiscono il disegno dell'icona
68	$80 + USR	Tipo C-64 del file utilizzato quando il file e' salvato da GEOS. PRG=1, SEQ=2, USR=3, REL=4. Bit 6=1 indica che il file e' protetto in scrittura
69	6	Tipo GEOS del file (vedere la tavola precedente)
70	0	Struttura GEOS del file: 0=SEQUENTIAL, 1=VLIR
71 - 72	FileStart	Indirizzo in memoria dove il file inizia a essere memorizzato
73 - 74	FileEnd	Indirizzo in memoria dell'ultimo byte del file memorizzato
75 - 76	InitProg	Indirizzo della routine di inizializzazione da eseguire dopo il caricamento
77 - 96	PermanentString	Solo i primi 16 dei 20 byte sono utilizzabili. Byte 0 - 11 nome permanente completato con spazi. Byte 12 - 15 stringa versione, es: V1.3. Byte 16 - 20 azzerati
97 - 116	ParentDisk	Se il file contiene i dati di un'applicazione, sono disponibili 20 byte per il nome del disco che contiene l'applicazione.
	Nome autore	Se il file e' un'applicazione, sono disponibili 20 byte per il nome dell'autore

SEGUE

SEGUE

Byte	Contenuto	Descrizione
117 - 136	ParentApplication	Se il file contiene i dati di un'applicazione, sono disponibili 20 byte per il nome dell'applicazione. Byte 0 - 11 nome dell'applicazione completato con spazi. Byte 12 - 15 stringa versione, es: V1.3. Byte 16 - 20 azzerato
137 - 159	Application	23 byte disponibili per le applicazioni
160 - 255	Info	Questo spazio viene impiegato dall'opzione Info del menu file presente in deskTop per memorizzare i commenti al file digitati dall'utente. Il testo dev'essere una stringa a terminazione nulla. Se non e' presente alcun testo il primo byte (il byte in posizione 160) dev'essere azzerato

I byte 2 e 3 contengono la larghezza e l'altezza del disegno che rappresenta l'icona. I dati che generano il disegno sono situati dal byte 4 al byte 67. Le icone dei file hanno sempre la stessa dimensione: 3 byte di larghezza e 21 linee di scansione in altezza. Le dimensioni dell'icona precedono i dati del disegno in quanto la routine interna di visualizzazione delle icone è generalmente in grado di visualizzare icone di qualunque dimensione, e prevede che le dimensioni dell'icona precedano i dati del disegno. Lo spazio dal byte 4 al byte 67 contiene il disegno dell'icona memorizzato in formato compattato Bit-Map. Il byte 4 è il byte di formato che identifica il tipo di compattazione. GEOS prevede tre formati differenti di compattazione. Il secondo formato, come già illustrato nel capitolo dedicato alla grafica, indica un gruppo di dati Bit-Map espansi dopo la compattazione. Quando si adotta questo formato, il byte 4 dev'essere un numero compreso fra 128 e 220. Il numero di byte interessati dal formato di compattazione si ottiene sottraendo 128 al byte di formato. Dal momento che sottrarre 128 significa azzerare il bit 7, i bit rimanenti indicano il numero di byte che seguono. Questo numero dev'essere compreso fra 1 e 92. Un'icona delle dimensioni specificate è composta da 63 byte, e quindi il byte formato si ottiene con l'operazione $80 + 63.

I tre bit meno significativi del byte 68 indicano il tipo del file tra quelli riconosciuti dal 1541 (PRG, SEQ, USR, DEL). Il byte 69 specifica il tipo del file in ambiente GEOS. Attualmente GEOS mette a disposizione 15 differenti tipi di file. È possibile che in

futuro ne vengano resi disponibili altri, ma riguarderanno con ogni probabilità file di dati particolari che in questa sede raggrupperemo nel tipo generico APPL_DATA. Il byte 70 indica la struttura GEOS adottata per organizzare il file su disco. Abbiamo già accennato ai due tipi di struttura disponibili, VLIR e SEQUENTIAL; è opportuno ricordare ancora che la struttura SEQUENTIAL in ambiente GEOS individua esclusivamente tutti i file composti da un concatenamento di blocchi sequenziale, e non dev'essere confusa con il tipo SEQ caratteristico del disk drive 1541.

I byte 71 e 72 costituiscono una word che individua l'indirizzo in memoria al quale GEOS deve iniziare a memorizzare il file prelevato da disco. Di solito GEOS accede a questa informazione per caricare un file da disco in memoria, ma, come vedremo, può anche utilizzare un indirizzo alternativo se l'applicazione lo richiede espressamente. Questa seconda procedura si rende utile quando l'applicazione deve caricare un file dati da disco e lo deve allocare in memoria a un particolare indirizzo, che può anche essere diverso di volta in volta. I byte 73 e 74 costituiscono una word che individua l'indirizzo dell'ultimo byte del file in memoria, ovvero la fine del file. Questa è un'informazione utile per diversi scopi. Quando GEOS deve salvare un file su disco, per prima cosa calcola la lunghezza del file per accertarsi che il disco abbia lo spazio sufficiente per contenerlo; l'applicazione deve preoccuparsi di aggiornare la word che contiene l'indirizzo di fine file nel caso che la lunghezza del file in memoria venga alterata. Quando GEOS deve caricare in memoria un file di tipo ASSEMBLY o BASIC, il calcolo della lunghezza diventa fondamentale. Se il file non si sovrappone al Kernel, GEOS è in grado di caricarlo utilizzando il turbo, riducendo così considerevolmente il tempo di "loading" necessario. Se invece il file è talmente lungo da sovrapporsi ai codici del Kernel, GEOS lo carica utilizzando le normali (e lente) routine messe a disposizione dal Kernel del C-64.

I file eseguibili sono quelli di tipo BASIC, ASSEMBLY, APPLICATION, DESK_ACC e AUTO_EXEC. Dopo che il file è stato caricato in memoria, GEOS accede alla word individuata dai byte 75 e 76 per sapere qual è l'indirizzo della routine a cui cedere il controllo per avviare l'applicazione. Normalmente si tratta dell'indirizzo di caricamento specificato dai byte 71 e 72, in quanto di solito la routine d'inizializzazione è la prima dell'applicazione (ma non è certo una regola).

I successivi 20 byte del blocco memorizzano la stringa che indica il nome permanente del file (PermanentName). Sebbene siano allocati 20 byte per questa informazione, gli ultimi 4 dovrebbero essere sempre a zero. La lunghezza massima di 16 caratteri imposta a questa stringa è necessaria per mantenere la compatibilità con la lunghezza dei nomi di file. Dei 16 byte significativi, i byte 0-11 sono effettivamente impiegati per il nome, eventualmente completato con alcuni spazi. Questo nome caratterizza tutti i file dati generati dalla stessa applicazione. Quando l'applicazione scorre la directory di un disco per raggruppare tutti i file da essa generati, accede al PermanentName di ogni file di tipo APPL_DATA e lo confronta con il nome permanente assegnato ai suoi file dati. Oltre al nome permanente, è importante anche sapere qual è il formato usato per memorizzare i dati nel file. Nei byte 12-15 è indicata

appunto la particolare versione di questo formato. Per una nostra convenzione, l'identificatore è rappresentato da una V maiuscola seguita da due numeri separati da un punto: V1.0. Il primo dei due numeri è il più significativo. Si noti che la versione del *formato* dei file dati non ha alcuna relazione con la versione dell'*applicazione*.

Per un file di dati è assolutamente necessario un nome permanente di qualche tipo, grazie al quale l'utente sia libero di dare al file un nome significativo e in seguito cambiarlo. Per fare un esempio, geoWrite dev'essere capace di riconoscere che un file di dati versione 1.0 è effettivamente un file di dati geoWrite V1.0, nonostante il nome assegnatogli.

A seguito dei 20 byte allocati per il nome permanente si trovano altre due stringhe da 20 byte che possono essere utilizzate dalle applicazioni in vari modi. Come accade per il nome permanente, anche queste due stringhe contengono un nome nei byte 0-11 e la versione nei quattro successivi. Gli ultimi quattro byte non vengono utilizzati. La prima delle due stringhe, allocata nei byte 97-112, è denominata ParentDisk, e contiene il nome del disco sul quale si trova l'applicazione che ha generato il file. Se però il file non è un file dati, ma un'applicazione, in questa stringa può apparire il nome del suo autore. Attualmente il nome ParentDisk non viene impiegato dalle applicazioni GEOS compatibili.

Quando l'utente preme due volte in rapida successione il pulsante del mouse in corrispondenza dell'icona di un file dati, deskTop accede alla stringa ParentApplication, allocata nei byte 117-132, per sapere qual è il nome dell'applicazione che ha generato il file. Se non trova l'applicazione sullo stesso disco, visualizza un messaggio all'utente nel quale richiede che sia inserito il disco opportuno, come per esempio: "Please insert a disk with geoWrite". Quando GEOS scorre la directory di un disco alla ricerca dell'applicazione il cui nome è specificato nella stringa ParentApplication, confronta i nomi dei file solo per i primi 12 byte, e ignora l'informazione sulla versione contenuta nei byte successivi. Spetta all'utente controllare che la versione sia quella che GEOS si aspetta. Vi è la tacita convenzione che le applicazioni devono essere in grado di gestire tutti i file dati generati da versioni precedenti della stessa applicazione. Tutti i programmatori dovrebbero tenerne conto, durante la modifica delle loro applicazioni.

Quando l'applicazione viene caricata ed eseguita, deve accedere alla stringa del PermanentName assegnata al file dati che sta analizzando. Di norma questa stringa è la stessa utilizzata per il nome ParentApplication, anche se il numero della versione può essere diverso. Se per esempio l'utente seleziona da deskTop un file dati creato con l'applicazione geoWrite V1.2 e inserisce un disco contenente l'applicazione geoWrite V2.0, deskTop, che non confronta i numeri di versione, carica ed esegue l'applicazione geoWrite V2.0. È compito dell'applicazione, di geoWrite, accedere al nome permanente del file dati e decidere se si rende necessaria la conversione dei dati nel formato adottato dalla nuova versione. Se le due versioni 1.2 e 2.0 presentano differenze sostanziali, geoWrite V2.0 dev'essere in grado di effettuare le opportune conversioni per essere compatibile con i dati creati dalla versione precedente.

Di solito la versione di un programma "cresce" più rapidamente della versione del formato dei dati che esso impiega; questo accade perché di un programma si realizzano nuove versioni anche solo per correggere tutti gli errori riscontrati. I numeri di versione che individuano i formati dei dati tendono a seguire aggiornamenti più discontinui di quelli delle applicazioni. Per fare un esempio, una particolare applicazione viene introdotta sul mercato come versione 1.0. I dati da essa generati mantengono lo stesso numero di versione 1.0. Dopo un mese di esperimenti viene prodotta la versione 1.1 dell'applicazione. Dopo neanche due settimane di ricerche viene scovato e corretto un altro bug, e la versione 1.1 è sostituita dalla 1.2. Durante tutto questo processo di revisione, il formato dei dati è rimasto inalterato e corrisponde ancora alla versione 1.0. Tutte le versioni possono ancora accedere ai dati senza dover operare conversioni. Sei mesi più tardi viene introdotta sul mercato la versione 2.0 dell'applicazione, che presenta sostanziali differenze dalle versioni precedenti, anche nel formato dei dati. La versione del formato dei dati è quindi aggiornata alla V2.0, saltando il processo di revisione che ha portato alla creazione delle V1.1 e V1.2. Il cambio della versione dei dati serve per indicare alle applicazioni con versione precedente che non possono gestire il nuovo formato dei dati. Se l'utente dispone dell'ultima versione non dovrebbe più utilizzare le versioni precedenti.

È compito dell'applicazione realizzare (all'interno dei suoi codici d'inizializzazione) alcune routine con lo scopo di controllare la versione dei dati per determinare se l'applicazione li può gestire. Nel caso che li possa gestire, l'applicazione deve determinare se i dati contenuti nel file devono essere trasformati nel formato corrente.

Esempio d'impiego del nome permanente

Supponiamo che l'utente selezioni l'icona corrispondente a un documento generato da geoWrite V1.0; deskTop inizia a cercare lungo la directory un file che contenga nella stringa ParentApplication il nome "geoWrite". Nella sua ricerca, deskTop non tiene conto della versione, ma confronta solo i primi 12 caratteri dei nomi. Se sul disco non è presente alcun file con il nome richiesto, deskTop visualizza un messaggio per l'utente nel quale richiede l'inserimento di un disco che contenga un file con quel nome. La versione dell'applicazione non compare mai nel nome dell'applicazione. Quando deskTop identifica il file geoWrite su disco, comanda a GEOS di caricarlo ed eseguirlo. Prima di farlo, deskTop aggiorna alcuni flag e inserisce una stringa che contiene il nome del file dati che sarà elaborato da geoWrite. L'applicazione, in questo caso geoWrite, riceve il controllo e dispone delle informazioni lasciate da deskTop relative al documento da elaborare. Con queste informazioni geoWrite localizza il file dati, accede al PermanentName a esso associato e legge la versione del formato dei dati memorizzati. Quindi determina se è in grado di leggere i dati direttamente, o se deve trasformarli in un formato adatto prima di procedere all'elaborazione. Può anche accadere che la versione in esecuzione di geoWrite sia la V1.0 e il file dati sia stato

generato con la versione V2.0. In questo caso geoWrite non può elaborare il file e termina la propria esecuzione, restituendo il controllo a deskTop, o chiedendo l'inserimento di un altro disco.

Le costanti per accedere al sistema dei file

Tutte le costanti riportate nelle tavole precedenti sono state introdotte per facilitare il lavoro del programmatore, e sono raccolte organicamente nel file geosConstants in appendice. I nomi delle costanti sono studiati espressamente per rendere più comprensibile il codice sorgente e aiutare il programmatore a ricordare le posizioni dei byte significativi e i valori che determinati byte possono assumere. La maggior parte delle costanti si utilizza per puntare particolari byte riportati nelle tavole precedenti. Il file geosConstants è suddiviso nelle seguenti sezioni: Tipi dei file in ambiente GEOS, Tipi Commodore dei file, Intestazione della directory, File Entry, File Header e Costanti per il disco.

Le variabili per l'accesso al disco

Quando GEOS si prepara a caricare ed eseguire un'applicazione, ripone in memoria una serie di informazioni. Altre informazioni sono mantenute abitualmente in memoria da deskTop per suo uso interno: sia le une che le altre sono disponibili all'applicazione durante la sua esecuzione.

Infine, sempre durante la fase preliminare, deskTop imposta una serie di variabili che costituisce il gruppo d'informazioni di prima necessità per l'applicazione. I dati per la gestione degli accessori da scrivania (Desk Accessory) sono sostanzialmente diversi. Maggiori dettagli per la gestione degli accessori da scrivania si troveranno nei prossimi capitoli, nelle descrizioni delle routine GetFile e LdDeskAcc.

L'applicazione può disporre di diverse variabili create appositamente per comunicare con il drive. La variabile curDrive indica il numero del drive che contiene il disco sul quale è memorizzata l'applicazione (può essere l'8 o il 9). La variabile curDevice è allocata all'indirizzo $00BA, dove il C-64 conserva il numero del dispositivo attualmente in funzione. All'inizio dell'esecuzione di un'applicazione queste due variabili contengono lo stesso valore. Infine i byte Disk ID del disco che contiene l'applicazione vengono memorizzati nella memoria interna del drive.

Durante il processo di caricamento di un'applicazione vengono aggiornate numerose variabili. Supponiamo per esempio che l'utente abbia selezionato un file dati: GEOS carica in memoria l'applicazione che l'ha creato e le passa il nome del file dati in modo che questa sappia automaticamente quale file deve elaborare. Un particolare bit in r0L viene impostato a 1 per indicare che è stato specificato il nome di un file dati. In questo caso, r3 punta al nome del file dati e r2 punta al nome del disco sul quale è

memorizzato. Un'applicazione può essere richiamata anche solo per stampare un file dati. In questo caso un altro bit in r0L indica all'applicazione quali sono le operazioni da compiere con il file.

r0L flag loadOpt

Bit 1 (solo per i file eseguibili)
- 0 non è specificato alcun file dati da elaborare automaticamente
- 1 è specificato un file dati selezionato dall'utente e l'applicazione deve elaborarlo automaticamente (la costante per questo bit è ST_LD_DATA).

Bit 6 (solo per i file eseguibili)
- 0 il file dati non è stato selezionato per essere stampato
- 1 deskTop imposta a 1 questo bit quando l'utente preme una sola volta l'icona di un file dati e seleziona l'opzione di stampa. L'applicazione deve stampare il file e restituire il controllo a deskTop (la costante per questo bit è ST_PR_DATA).

I registri r2 e r3 contengono dati significativi solo se il bit 1 ed eventualmente il bit 6 in r0L sono impostati a 1.

r2 Puntatore al nome del disco che contiene il file dati. Normalmente r2 punta al buffer dataDiskName che contiene il nome del disco sul quale si dovrebbe trovare il file dati da elaborare automaticamente. L'applicazione, una volta individuato il file e operati i relativi controlli di compatibilità, lo elabora secondo quanto specificato dal bit 6 in r0L.

r3 Puntatore al nome del file dati. Di solito r3 indica il buffer dataFileName, che contiene il nome del file dati che l'applicazione deve elaborare automaticamente.

Nelle variabili globali di GEOS sono memorizzati anche il File Entry, il blocco File Header del file, e il Directory Header Block del disco.

dirEntryBuf	File Entry del file
curDirHead	Directory Header Block del disco che contiene il file
fileHeader	Blocco File Header del file, che GEOS associa a tutti i suoi file

GEOS memorizza anche una tavola, durante il caricamento del file, che contiene gli indirizzi T/S di tutti i blocchi che compongono il file caricato sequenzialmente dal disco. Questa tavola è denominata fileTrScTab e occupa un blocco in memoria.

fileTrScTab Lista degli indirizzi T/S dei blocchi del file. Il numero massimo di blocchi di cui un file può essere composto è 127 (32.258 byte).

La prima word della tavola fileTrScTab è l'indirizzo T/S del blocco File Header associato al file. Le 127 word seguenti indicano gli indirizzi T/S dei restanti blocchi del file.

r5L Offset dall'inizio della tavola fileTrScTab fino all'ultimo indirizzo T/S significativo. r5L punta la word della tavola che individua l'ultimo blocco del file.

Definite le variabili più importanti dedicate alla gestione dei file su disco, possiamo iniziare a introdurre le molteplici routine di cui GEOS dispone per comunicare con il drive. Il prossimo paragrafo è una panoramica su tutte le routine disponibili e sul modo di utilizzarle, e comprende anche un'introduzione su come devono essere effettuati gli accessi al bus seriale in ambiente GEOS.

Le routine di accesso al disco in ambiente GEOS

Il Kernel di GEOS mette a disposizione delle applicazioni una serie completa di routine di accesso al disco (disk routine). L'insieme di queste routine riesce a soddisfare una vastissima gamma di esigenze anche molto diverse fra loro, in quanto spazia da routine specifiche molto potenti a routine primitive molto generali. La maggior parte delle applicazioni utilizzano solo le routine più complesse, che svolgono molte operazioni e sono in grado di soddisfare solo le esigenze più comuni. Routine di questo genere vengono chiamate "d'alto livello" in quanto liberano il programmatore da buona parte dei problemi secondari. Altre applicazione si trovano a dover soddisfare esigenze più particolari, e le routine d'alto livello non sono abbastanza flessibili per adattarsi a questi casi. GEOS mette a disposizione a questo scopo una serie di routine di livello intermedio che l'applicazione può facilmente combinare insieme per ottenere funzioni particolari che non possono essere realizzate con le routine d'alto livello. D'altra parte le stesse routine d'alto livello sono combinazioni di routine di livello intermedio.

Infine, GEOS possiede un livello primitivo di routine per l'accesso al disco, raramente utilizzate dalle applicazioni. Esse svolgono operazioni semplici ma specifiche, in stretto contatto con l'hardware del computer. Se un'applicazione deve comunicare con uno speciale dispositivo esterno diverso dalla stampante e dal drive, queste routine diventano molto utili. Alcune per esempio permettono all'applicazione di chiamare le routine presenti nel Kernel del C-64 e nel DOS del drive. Negli stadi finali tutte le routine presenti nei livelli superiori si servono delle routine di livello primitivo per comunicare con il drive.

Nozioni di base per accedere al disco

In ambiente GEOS solo un dispositivo esterno alla volta può comunicare attraverso il bus seriale. Normalmente è uno dei due drive, A o B, o la stampante, ma in linea di massima potrebbe essere qualunque tipo di dispositivo. Per cambiare il dispositivo che correntemente impegna il bus seriale viene utilizzata la routine SetDevice. Quando un dispositivo impegna il bus seriale si dice "attivato". L'applicazione deve passare a SetDevice il numero del dispositivo che si vuole attivare, per esempio 8 o 9 per i due drive.

Se il dispositivo selezionato è un drive, l'operazione successiva è quella di chiamare la routine OpenDisk per iniziare la procedura di accesso al disco. OpenDisk inizializza il drive e alcune variabili che il Kernel di GEOS impiega per l'accesso al disco.

Una volta che la comunicazione con il drive è stata avviata, l'applicazione può chiamare una delle seguenti routine.

Routine d'alto livello

SetDevice	- Cede il bus seriale al dispositivo prescelto.
OpenDisk	- Prepara il sistema ad accedere al disco correntemente inserito.
GetPtrCurDkNm	- Restituisce un puntatore che individua il buffer contenente il nome del disco.
SetGEOSDisk	- Converte un normale disco utilizzato in modo C-64 in un disco in formato GEOS, allocando il blocco Off Page per la directory.
ChkDkGEOS	- Controlla se il disco correntemente attivato è in formato GEOS.
FindFTypes	- Genera una lista contenente tutti i file presenti su disco appartenenti a un tipo particolare.
GetFile	- Passando il nome di un file, la routine lo carica in memoria. Si può trattare di un file dati, di un'applicazione o di un accessorio da scrivania. Se il file è eseguibile, GetFile lo esegue.
FindFile	- Scorre la directory, compreso il blocco Off Page, alla ricerca del file specificato. Restituisce il File Entry.
SaveFile	- Salva una zona di memoria in un file GEOS su disco.
DeleteFile	- Cancella un file da disco.
RenameFile	- Cambia il nome del file con uno nuovo.
EnterDeskTop	- Termina l'esecuzione dell'applicazione e restituisce il controllo a deskTop.
CalcBlksFree	- Calcola quanti blocchi sono ancora disponibili sul disco.
Routine VLIR	- Vedere il capitolo dedicato alla struttura VLIR.

Routine di livello intermedio

Le routine che abbiamo appena elencato non possono ovviamente soddisfare tutte le possibili esigenze. Per esempio, GEOS non fornisce alcuna routine in grado di formattare un disco o copiare un file. La maggior parte delle applicazioni hanno raramente bisogno di copiare dischi o file e quindi queste possibilità non sono state incluse nel Kernel di GEOS. Nonostante questo, deskTop prevede entrambe le funzioni. deskTop è un'applicazione come le altre (geoWrite o geoPaint per esempio), con la differenza che è dedicata alla manipolazione dei file. Le funzioni di copia, di formattazione e di convalida disponibili in deskTop sono state realizzate combinando opportunamente le routine di livello intermedio.

Il programmatore deve prestare particolare attenzione quando impiega le routine di livello intermedio, dal momento che queste si aspettano che l'applicazione abbia già compiuto determinate operazioni quando sono in esecuzione. Chiamandone una senza aver preventivamente inizializzato le variabili e/o le tavole necessarie, è possibile alterare accidentalmente i dati su disco, bloccare il sistema definitivamente o entrambe le cose. In particolare molte routine si aspettano che il Kernel mantenga in memoria la copia del blocco su disco nel quale è contenuta la Directory Header, e che i dati memorizzati in questo blocco, locato a curDirHead, siano corretti. Se un dato viene alterato, dopo l'aggiornamento è necessario riscrivere il blocco su disco. La lista che segue riporta i nomi delle routine di livello intermedio, con una breve descrizione, seguendo un ideale ordine di utilità decrescente.

GetBlock	- Legge un blocco del disco.
PutBlock	- Memorizza su disco un blocco ed effettua la verifica dei dati scritti.
GetFHdrInfo	- Passando il File Entry di un file restituisce il blocco File Header del file.
ReadFile	- Legge da disco una serie di blocchi concatenati sequenzialmente.
WriteFile	- Trasferisce il contenuto di una zona di memoria in una serie di blocchi concatenati sequenzialmente su disco.
ReadByte	- Simula la lettura di un byte alla volta, da un gruppo di blocchi concatenati sequenzialmente su disco.
GetDirHead	- Legge dal disco la Directory Header e la BAM.
PutDirHead	- Memorizza sul disco la Directory Header e la BAM.
NewDisk	- Inizializza il drive per accedere a un nuovo disco senza alterare le variabili GEOS.
LdApplic	- Carica ed esegue un'applicazione GEOS compatibile.
LdFile	- Carica un file GEOS.
LdDeskAcc	- Carica ed esegue un accessorio da scrivania GEOS compatibile.

RstrAppl	- Routine eseguita dai DA quando terminano la loro esecuzione.
GetFreeDirBlk	- Individua lo spazio libero nella directory per un nuovo File Entry. Se necessario alloca un nuovo blocco per la directory.
BlkAlloc, NxtBlkAlloc	- Allocano un concatenamento di blocchi sul disco.
SetNextFree	- Alloca il primo blocco libero che incontra sul disco.
FindBAMBit	- Restituisce lo stato di un blocco su disco (allocato o libero).
FreeBlock	- Disalloca un blocco particolare su disco.
SetGDirEntry, BldGDirEntry	- Crea in memoria e su disco il File Entry di un file.
FollowChain	- Crea una lista di indirizzi T/S di blocchi appartenenti a un concatenamento e la memorizza nella tavola a fileTrScTab.
FastDelFile	- Disalloca tutti i blocchi indicati dalla lista di indirizzi T/S memorizzata nella tavola a fileTrScTab.
FreeFile	- Disalloca tutti i blocchi di un file. Lascia il File Entry intatto.
ChangeDiskDevice	- Cambia il numero di device (8 o 9) a cui corrisponde il drive.
StartAppl	- Riconfigura il sistema a una situazione di default.

Routine di livello primitivo

GEOS mette a disposizione anche un gruppo di routine di livello primitivo. Possono esserci solo tre ragioni per impiegare queste routine:

1) per accedere alle routine contenute nel Kernel del C-64. deskTop accede alle routine del Kernel del C-64 per formattare il disco
2) per comunicare con un dispositivo diverso da un drive o una stampante
3) per realizzare routine di accesso al disco particolarmente ottimizzate, in grado di muovere un grande numero di blocchi organizzati su disco in qualche maniera insolita. Questa possibile applicazione rappresenta un caso estremo e di solito le routine per leggere e scrivere i dati raggruppati in blocchi concatenati sono sempre sufficienti.

Ecco quali sono le operazioni non realizzabili direttamente tramite GEOS, per le quali un'applicazione potrebbe avere la necessità di accedere al bus seriale. Le routine che elenchiamo sono state realizzate per offrire accessi sicuri al bus seriale e ritorni altrettanto sicuri al sistema dei file gestito da GEOS.

InitForIO	- Disabilita tutti gli interrupt, disattiva gli sprite, configura i banchi di memoria in modo che siano accessibili il Kernel del C-64 e lo spazio di I/O.
DoneWithIO	- Ripristina tutti gli interrupt, riattiva gli sprite, configura i banchi di memoria in modo che tutta la RAM disponibile sia accessibile (la normale situazione in ambiente GEOS).
PurgeTurbo	- Normalmente il codice del turbo è sempre in esecuzione. PurgeTurbo rimuove il codice del turbo memorizzato nella RAM del drive 1541 e restituisce il controllo del bus seriale al DOS.
EnterTurbo	- Attiva, caricandolo se è stato rimosso, il codice del turbo presente nel drive.
ExitTurbo	- Disattiva il codice del turbo nel drive, ma non lo disalloca.
ReadBlock	- Legge un blocco da disco. Il codice del turbo residente nel drive dev'essere già in esecuzione, e la routine InitForIO dev'essere già stata chiamata.
WriteBlock	- Memorizza un blocco su disco. Non viene eseguita alcuna verifica, il codice del turbo dev'essere in esecuzione e la routine InitForIO dev'essere già stata eseguita.
VerWriteBlock	- Come WriteBlock, e inoltre verifica che il trasferimento dei dati sia avvenuto in maniera corretta.

Le comunicazioni con il bus seriale

Ecco la procedura da adottare per accedere con successo al bus seriale del C-64:

1) chiamare la routine SetDevice per attivare le comunicazioni con il dispositivo prescelto. SetDevice cede il bus seriale al dispositivo
2) se si desidera utilizzare le routine rese disponibili dal DOS del 1541, si deve disattivare il codice del turbo presente nel drive tramite la routine PurgeTurbo. Se non si deve accedere alle routine del Kernel del C-64, questo punto è da saltare
3) chiamare InitForIO per disabilitare gli interrupt, gli sprite e abilitare lo spazio di I/O e il Kernel del C-64
4) chiamare le routine del Kernel del C-64 per accedere al bus seriale e comunicare con dispositivi esterni
5) quando si è finito di utilizzare il bus seriale, si deve chiamare DoneWithIO. Questa routine ripristina la configurazione di sistema precedente alla chiamata di InitForIO. La successiva disk routine di GEOS che l'applicazione chiama (eccettuate ReadBlock, WriteBlock o VerWriteBlock) riattiva automaticamente il turbo di accesso al disco.

La trasformazione dei file normali in file GEOS

Trasformare un file normale, di qualsiasi forma e struttura, in un file GEOS, significa dotarlo del blocco File Header, e quindi di tutte le informazioni in esso contenute, ed espanderne il File Entry. Queste operazioni possono essere compiute indifferentemente su file Basic, file in Assembly e file dati che non sono stati creati per interagire con l'ambiente GEOS.

Per effettuare la trasformazione, è sufficiente richiedere all'applicazione Icon Editor di cambiare l'icona del file interessato. L'applicazione è in grado di determinare se il file è in formato GEOS o meno, e in questo secondo caso chiede all'utente le informazioni necessarie per procedere alla trasformazione. Prima di farlo, è comunque buona norma effettuare una copia di salvataggio del file, per non rischiare di perderlo se nascono problemi.

14 ROUTINE D'ALTO LIVELLO

Nel sistema operativo GEOS sono disponibili molte routine per accedere ai file e manipolarli. Questa collezione di funzioni spazia dalle routine d'alto livello a quelle più elementari che costituiscono la struttura di base delle routine di livello superiore. Nel capitolo vengono illustrate le seguenti routine.

SetDevice
OpenDisk
GetPtrCurDkNm
SetGEOSDisk
ChkDkGEOS
FindFTypes
GetFile
FindFile
SaveFile
DeleteFile
RenameFile
EnterDeskTop
CalcBlksFree

Prima di iniziarne la descrizione, ricordiamo che tutte le routine restituiscono nel registro x il codice d'errore da disco. Questo codice informa sull'esito del tentativo d'accesso al disco (positivo x = 0, negativo x ≠ 0).

Nel caso che x sia diverso da zero, il valore restituito identifica il tipo di errore. Per una consultazione dei possibili errori da disco gestiti da GEOS si veda l'appendice.

SetDevice

Funzione: Cede il bus seriale al dispositivo indicato.

Indirizzo: $C2B0

Parametri:

a	numero del dispositivo (8 - 11 per il disk drive, 4 per la stampante)
curDevice	dovrebbe già essere impostato per il dispositivo corrente

Restituisce:

curDevice	numero del nuovo dispositivo, come passato in a
curDrive	se il nuovo dispositivo è un disk drive (8 - 11) il suo numero di device è memorizzato in curDrive
x	codice d'errore di accesso al disco (vedere i codici d'errore riportati in appendice)

Distrugge: a, x, y

Sinossi: È necessario impiegare SetDevice quando si desidera cambiare il dispositivo che sta utilizzando il bus seriale. Se il dispositivo corrente è un disk drive, GEOS effettua una chiamata a ExitTurbo chiedendo al turbo in esso memorizzato di cedere al DOS il controllo del bus seriale. Il turbo presente nella RAM del drive non viene cancellato, ma il drive si configura in modo DOS 1541. Se si desidera cedere il bus seriale alla stampante bisogna passare come numero di device il 4.

OpenDisk

Funzione: Inizializza il drive per accedere a un nuovo disco. Dev'essere già stata eseguita la routine SetDevice.

Indirizzo: $C2A1

Chiama: NewDisk, GetDirHead, ChkDkGEOS, GetPtrCurDkNm

Parametri:

curDrive	dovrebbe già essere stata impostata da SetDevice per indicare il drive desiderato

Restituisce:

Dr?CurDkNm	memorizza in questo buffer il nome del disco. Il carattere "?" presente nel nome della variabile, dev'essere sostituito con i caratteri A o B, per indicare se si tratta del buffer del drive A o del drive B. L'indirizzo del buffer si ottiene chiamando la routine GetPtrCurDkNm
r5	punta a Dr?CurDkNm. Viene ottenuto chiamando la routine GetPtrCurDkNm
curDirHead	il Directory Header Block del disco, ottenuto chiamando la routine GetDirHead
isGEOS	se il disco è in formato GEOS questo flag è impostato a TRUE. Viene ottenuto chiamando la routine ChkDkGEOS
x	codice d'errore (0 se non c'è nessun errore. Vedere l'appendice per i vari codici d'errore)

Distrugge: a, x, y, r0 - r4, r6 - r15

Sinossi: OpenDisk inizializza l'accesso al disco inserito nel drive corrente. Si ricorre a OpenDisk quando l'utente inserisce un nuovo disco, dopo l'esecuzione di SetDevice per cedere il bus seriale al drive corrente. La routine chiama NewDisk per preparare il drive al nuovo disco. In seguito OpenDisk chiama la routine GetDirHead per leggere il Directory Header Block e memorizzarlo nel buffer curDirHead. Poi chiama ChkDkGEOS e imposta a TRUE il flag isGEOS se il disco è in formato GEOS. Dopo questa operazione, chiama GetPtrCurDkNm per leggere il nome del disco (la stessa stringa che compare sotto l'icona del drive) e memorizzarlo nel buffer Dr?CurDkNm (il carattere "?" dev'essere sostituito da A o da B per indicare il drive corrente). Questo buffer è puntato dal registro r5. Se il codice turbo non è attivo, la routine lo installa e lo lascia in esecuzione per eventuali nuovi accessi al disco.

GetPtrCurDkNm

Funzione: Restituisce un puntatore a uno dei due buffer DrACurDkNm e DrBCurDkNm. Il buffer puntato contiene il nome del disco presente nel drive correntemente selezionato.

Indirizzo: $C298

Chiamata da: OpenDisk

Chiama: BBMult

Parametri:

x	indirizzo del registro in pagina 0 nel quale la routine deve restituire l'indirizzo del buffer che contiene il nome del disco. Per esempio: ldx #r8 significa che la routine utilizza r8. Il registro r15 non può essere impiegato
curDrive	numero del disk drive correntemente selezionato, di solito 8 o 9
DrACurDkNm	contiene il nome del disco nel drive A
DrBCurDkNm	contiene il nome del disco nel drive B

Restituisce: Il puntatore a Dr?CurDkNm memorizzato nel registro specificato da x

Distrugge: a, y, r15

Sinossi: GetPtrCurDkNm utilizza la variabile curDrive per determinare quale dei due drive è correntemente selezionato (drive A o B). Quindi restituisce nel registro puntato da x l'indirizzo del buffer DrACurDkNm o DrBCurDkNm contenente il nome del disco. Per utilizzare questa routine è sufficiente passare in x l'indirizzo del registro che si desidera impiegare; per esempio, se l'applicazione esegue l'istruzione ldx #r8, la routine restituisce il puntatore nel registro r8. Può essere impiegato qualsiasi registro, eccetto r15 che viene utilizzato internamente. Il nome per il disco corrente deve già essere memorizzato in DrACurDkNm o DrBCurDkNm. Il registro specificato viene restituito come puntatore al buffer Dr?CurDkNm.

SetGEOSDisk

Funzione: Scrive la stringa d'identificazione GEOS ID all'interno della Directory Header del disco, alloca il blocco Off Page della directory, riscrive nel disco la Directory Header e la BAM. In questo modo il disco viene convertito nel formato GEOS.

Indirizzo: $C1EA

Chiama: GetDirHead, CalcBlksFree, SetNextFree, PutDirHead

Parametri: curDrive numero del drive corrente (8 o 9)

Restituisce: curDirHead contiene la stringa GEOS ID e il puntatore T/S al blocco Off Page della directory
disco la Directory Header aggiornata e il blocco Off Page allocato

Distrugge: a, x, y

Sinossi: SetGEOSDisk converte un normale disco, già formattato e utilizzabile in ambiente C-64, in un disco in formato GEOS, salvaguardando tutti i dati presenti. Il formato GEOS si ottiene aggiungendo al disco la stringa GEOS ID e allocando il nuovo blocco Off Page della directory. SetGEOSDisk chiama la routine GetDirHead per leggere la Directory Header del disco e memorizzarla nella variabile curDirHead. Nello spazio della Directory Header normalmente inutilizzato, la routine memorizza la stringa GEOS ID d'identificazione del formato GEOS. In seguito alloca un settore del disco (tramite CalcBlksFree e SetNextFree) per contenere il blocco Off Page, e memorizza l'indirizzo T/S a questo settore nei byte 171 e 172 del Directory Header Block. SetDevice dev'essere già stata eseguita quando viene chiamata SetGEOSDisk, in modo che la variabile curDevice contenga il numero corretto del drive e quest'ultimo abbia accesso al bus seriale.

ChkDkGEOS

Funzione: Cerca la stringa GEOS ID nel disco corrente per individuare se è in formato GEOS.

Indirizzo: $C1DE

Chiamata da: OpenDisk

Parametri:

r5	puntatore all'immagine in memoria del Directory Header Block, memorizzato di solito a curDirHead
Directory Header	memorizzata di solito all'interno del buffer curDirHead

Restituisce:

isGEOS	TRUE se il disco è in formato GEOS
a	lo stesso valore memorizzato in isGEOS

Distrugge: x, y

Sinossi: ChkDkGEOS controlla se il disco inserito nel drive è in formato GEOS. Il controllo consiste nella lettura della stringa GEOS ID, contenuta nella Directory Header del disco, che dovrebbe essere stata opportunamente memorizzata nel buffer puntato da r5. Se questa stringa è presente, il flag isGEOS è impostato a TRUE. Si noti che il buffer puntato da r5 deve contenere l'intero Directory Header Block; ChkDkGEOS provvede poi a individuare all'interno di questo blocco la Directory Header.

FindFTypes

Funzione: Restituisce i nomi dei file caratterizzati dal tipo GEOS specificato.

Indirizzo: $C23B

Chiamata da: La routine di gestione del comando DBGETFILES per i box di dialogo

Parametri:

- r6 puntatore al buffer destinato a contenere i nomi dei file. Ogni nome è composto da un massimo di 16 caratteri più lo 0 come diciassettesimo
- r7L tipo GEOS dei file da cercare
- r7H massimo numero di nomi di file da inserire nella lista
- r10 0, per ignorare il nome permanente, oppure un puntatore alla stringa a terminazione nulla (lunga non più di 16 caratteri) che dev'essere confrontata con i nomi permanenti associati ai file. Questa stringa, facoltativa, rappresenta la seconda chiave di ricerca utilizzata da FindFTypes. In questo modo, non solo si possono raccogliere tutti i file di uno stesso tipo, ma si possono addirittura individuare all'interno del tipo quelli generati dalla stessa applicazione

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce: Il buffer viene riempito con i nomi dei file presenti sul disco che rispondono alle chiavi (o alla chiave se r10 = 0) di ricerca

Distrugge: a, y, r0, r1, r2L, r4, r6

Sinossi: FindFTypes costruisce una lista di nomi corrispondenti a file dello stesso tipo GEOS specificato in r7L. L'applicazione deve passare il tipo GEOS del file, un puntatore al buffer entro il quale la routine memorizza i nomi dei file, e il massimo numero di nomi che il buffer può contenere. I nomi dei file sono memorizzati in stringhe a terminazione nulla e ogni nome è completato da zeri sino al diciassettesimo carattere. Il diciassettesimo carattere è sempre uno 0, che costituisce il terminatore della stringa. I nomi dei file conservano nella lista lo stesso ordine con il quale appaiono sul disco, che è anche l'ordine con cui appaiono sul notes di deskTop. I tipi correnti di file sono: NOT_GEOS, BASIC, ASSEMBLY, DATA, SYSTEM, DESK_ACC, APPLICATION, APPL_DATA, FONT, PRINTER, INPUT_DEVICE, DISK_DEVICE, SYSTEM_BOOT, TEMPORARY, AUTO_EXEC. Per le singole spiegazioni, consultare l'appendice.

Facciamo un esempio che chiarisca la funzione svolta da questa routine. Quando si manda in esecuzione un'applicazione, si utilizza la routine FindFTypes per costruire la lista degli accessori da scrivania disponibili sul disco corrente (vengono presi in considerazione solo i primi otto). L'applicazione passa a FindFTypes il tipo DESK_ACC del file, il numero dei file significativi per la ricerca (MAX_DESK_ACC=8), e il puntatore al buffer allocato appositamente.

Prima che siano memorizzati i nomi dei file, il buffer viene interamente azzerato. In questo modo uno 0 come primo carattere di un nome indica la fine della lista. Durante il raggruppamento dei nomi, ogni volta che la routine incontra un file su disco che corrisponde alle caratteristiche specificate, r7H viene decrementato di una unità. Questo registro può essere utilizzato per calcolare velocemente il numero di file che la routine ha raggruppato: massimo numero di nomi – r7H.

La routine presenta la possibilità di specificare un'ulteriore chiave di ricerca. Questa permette di raggruppare file dello stesso tipo GEOS, per esempio APPL_DATA, e in più generati dalla stessa applicazione. FindFTypes confronta la seconda chiave di ricerca, realizzata da una stringa di caratteri, con il nome permanente di ogni file. Ricordiamo che il nome permanente di un file è memorizzato nel blocco File Header e identifica l'applicazione da cui proviene il file. Per fare un esempio, se geoPaint assegna ai file da essa creati il nome permanente "Paint Image V1.1", FindFtypes è in grado di individuarli e raggrupparli fra tutti i file di tipo APPL_DATA presenti sul disco. Il nome permanente è suddiviso in due parti contigue. La prima parte, dal carattere 1 al carattere 12, contiene il nome permanente che effettivamente viene confrontato con quello specificato dall'applicazione. La seconda parte, dal carattere 13 al carattere 16, contiene la versione del formato dei dati usato dall'applicazione e non viene presa in considerazione da FindFTypes. Spetta all'applicazione accedere alla

versione del formato e trarne le opportune conseguenze

Per riassumere, il nome permanente è composto da 16 byte significativi ed è memorizzato su disco in uno spazio di 20 byte. I quattro byte in più sono azzerati. Per introdurre questa seconda chiave di ricerca, l'applicazione deve specificare in r10 l'indirizzo della stringa a terminazione nulla che dev'essere confrontata con i nomi permanenti dei file che rispondono alla prima chiave di ricerca (tipo GEOS del file). GEOS confronta solo il numero di caratteri che compongono la stringa puntata da r10. In questo modo la stringa di ricerca non include il numero della versione, e non viene presa in considerazione la versione dell'applicazione che ha generato il file.

GetFile

Funzione: È la più importante routine di caricamento ed esecuzione dei file.

Indirizzo: $C208

Chiama: FindFile e una delle tre routine LdFile, LdDeskAcc o LdApplic

Parametri:

r6 punta alla stringa che contiene il nome del file. Questa stringa dev'essere a terminazione nulla e non più lunga di 16 caratteri, escluso il terminatore

r0L flag loadOpt

Bit 0
- 0 segue la procedura standard di caricamento
- 1 carica il file all'indirizzo specificato in r7 (la costante per questo bit è ST_LD_AT_ADDR)

Bit 6 (solo per file eseguibili)
- 0 il file non è stato selezionato per essere stampato direttamente
- 1 deskTop, per esempio, imposta a 1 questo bit quando l'utente muove l'icona di un file dati sopra l'icona della stampante e preme il pulsante del mouse. L'applicazione stampa il file e termina la sua esecuzione (la costante per questo bit è ST_PR_DATA)

Bit 7 (solo per file eseguibili)
- 0 non è specificato alcun file dati
- 1 bisogna caricare l'applicazione che ha creato il file selezionato dall'utente, perché il file sia automaticamente elaborato (la costante per questo bit è ST_LD_DATA)

r2 e r3 vengono passati solo se:
1) dev'essere caricata un'applicazione
2) i bit 6 e/o 7 in r0L sono impostati a 1

r2 puntatore al nome del disco che contiene il file dati selezionato che l'applicazione deve elaborare automaticamente. r2 e il contenuto del buffer dataDiskName sono dati importanti per l'applicazione. Questa, una volta che è stata caricata e mandata in esecuzione, elabora il file dati come specificato nel bit 6 di loadOpt

r3 punta al buffer che contiene il nome del file dati. L'applicazione utilizza questo puntatore per caricare il file dati in memoria ed elaborarlo

automaticamente; r3 e il contenuto del buffer dataFileName sono informazioni basilari per l'applicazione, che, dopo averle ricevute, elabora il file dati come specificato nel bit 6 di loadOpt

r7 contiene l'indirizzo al quale il file dev'essere memorizzato se il bit 0 in r0L, individuato da ST_LD_AT_ADDR, è impostato a 1

r10L flag di ripristino dei DA (Desk Accessory). Quando il file da caricare è un accessorio da scrivania, r10L contiene alcuni flag che specificano come dev'essere gestito

Bit 7	bit dello sfondo
0	GEOS non salva la parte di schermo coperta dalla finestra dell'accessorio da scrivania
1	GEOS salva la parte di schermo che viene nascosta e la ripristina al termine dell'esecuzione dei DA (la costante per questo bit è FG_SAVE)
Bit 6	bit del colore
0	non salva i colori
1	salva i colori interessati da cambiamenti e li ripristina dopo l'esecuzione dei DA (la costante per questo bit è CLR_SAVE)

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce: Restituisce il file in memoria all'indirizzo indicato nel blocco File Header del file o all'indirizzo specificato dal registro r7. Se il file è un'applicazione, o un accessorio da scrivania, viene mandato in esecuzione; il sistema viene inizializzato a una configurazione di default e l'esecuzione dell'applicazione inizia all'indirizzo specificato nel blocco File Header. Nel caso di un file eseguibile, GetFile non restituisce il controllo all'applicazione che l'aveva chiamato. r2, r3, r7 e r0L vengono passati all'applicazione o al DA. Nel caso di un'applicazione, r7 restituisce il vettore d'inizializzazione (indirizzo di inizio

esecuzione), prelevato dal File Header, invece dell'indirizzo al quale allocarlo in memoria. dataDiskName, loadAddr e dataFileName restano inalterati

x codice d'errore. Un qualunque errore causa la fine dell'operazione e la restituzione del controllo al codice che aveva effettuato la chiamata. Per quanto riguarda gli errori che si possono verificare nelle comunicazioni con il drive, vedere l'apposita sezione in appendice.

Suddividiamo ora i parametri restituiti dalla routine a seconda delle subroutine di GetFile da cui sono generati

Dalla chiamata alla routine FindFile:

dirEntryBuf File Entry del file

Dalla chiamata alla routine GetFHdrInfo attraverso una delle tre routine LdFile, LdApplic, LdDeskAcc (LdApplic a sua volta chiama LdFile):

fileHeader contiene il blocco File Header previsto dai file GEOS, anche se il file è a struttura VLIR

Dalla chiamata alla routine ReadFile attraverso una delle tre routine LdFile, LdApplic o LdDeskAcc:

fileTrScTab lista di indirizzi T/S dei blocchi del file o del record. La dimensione massima di un file è 127 blocchi (32.258 byte). GetFileHdrInfo memorizza nella prima word della tavola a fileTrScTab l'indirizzo T/S del blocco File Header del file. Quando viene eseguita GetFile per caricare un file in memoria, la routine ReadFile aggiorna la tavola a fileTrScTab. Se il file è a struttura VLIR, la tavola, a eccezione della prima word, contiene la lista degli indirizzi T/S dei blocchi di cui è composto il primo record del file

r1 nel caso che, durante il caricamento dei blocchi, raggiunga il limite imposto di 127 blocchi, la routine termina la lettura dei blocchi e la memorizzazione degli indirizzi T/S nella tavola, e il registro r1 contiene l'indirizzo T/S del blocco che non è stato letto. Questa informazione non viene restituita se viene eseguita la routine LdDeskAcc

r5L offset dall'inizio della tavola fileTrScTab all'ultimo indirizzo T/S dell'ultimo blocco del file, o del record

Dalla chiamata alla routine LdDeskAcc:

font — la fonte carattere selezionata è la fonte di sistema. In questo modo l'accessorio da scrivania inizia la sua esecuzione con la fonte di sistema

LdApplic e LdDeskAcc effettuano un'inizializzazione "a caldo" del sistema. Per sapere con precisione quali sono le variabili di sistema interessate da questa procedura, si veda la configurazione di sistema della partenza a caldo descritta nel capitolo 20

Distrugge: a, x, y, r0 - r10. I buffer dirEntryBuf e curDirHead non sono alterati

Sinossi: GetFile richiede solo il nome del file per caricare in memoria qualsiasi file GEOS compatibile. Se il file contiene dati, la routine lo carica alla locazione specificata nel blocco File Header del file (vedere la trattazione sulla struttura dei file). Se il file è un'applicazione o un DA, la routine lo carica in memoria all'indirizzo specificato nel blocco File Header del file e lo esegue.

Per eseguire queste operazioni, la routine deve per prima cosa cercare nella directory del disco il File Entry del file richiesto. Se il disco è in formato GEOS, la ricerca si estende anche al blocco Off Page della directory. Accedendo alle informazioni contenute nel File Entry, la routine è in grado di determinare il tipo GEOS del file e scegliere di conseguenza l'appropriata routine di caricamento. La procedura di caricamento dipende dal tipo GEOS del file. Per esempio, un file accessorio da scrivania richiede una procedura di caricamento sostanzialmente diversa da quella richiesta da un file dati.

Se l'utente seleziona un file dati e vuole che questo sia automaticamente elaborato dall'applicazione che l'ha generato, devono essere impostati gli appropriati bit nel flag loadOpt (r0L) in modo che GEOS sia in grado di svolgere questo compito. Per esempio, l'applicazione deskTop utilizza la routine GetFile nei seguenti casi:

1) l'applicazione è stata selezionata dall'utente con la doppia pressione del mouse sopra l'icona del file
2) l'utente ha selezionato il file dati prodotto da una particolare applicazione
3) l'utente ha richiesto la stampa di un file dati selezionandone l'icona e attivando l'opzione print del menu file.

Se deskTop richiede il caricamento di un'applicazione che inizi automaticamente l'elaborazione di un file dati, o che effettui la stampa del file e restituisca il controllo a deskTop (come avviene nel terzo caso sopra

citato), deve fornire all'applicazione le seguenti informazioni:

1) r2 deve puntare a una stringa in memoria che contenga il nome del disco sul quale è memorizzato il file dati
2) r3 deve puntare a una stringa in memoria che contenga il nome del file dati cha l'applicazione deve automaticamente elaborare.

Queste informazioni sono necessarie nel caso che deskTop debba cercare l'applicazione su un disco diverso da quello che contiene il file dati. Solo così l'applicazione, nonostante sia stata caricata su un altro disco, ha informazioni sufficienti per richiedere all'utente d'inserire il disco il cui nome è puntato da r2, e per iniziare a elaborare automaticamente il file dati originariamente selezionato. Se però i bit del flag loadOpt non indicano nessuno di questi casi, GetFile ignora le informazioni contenute nei registri r2 e r3, qualsiasi file debba caricare. In ogni caso il nome del file da caricare dev'essere puntato dal registro r6.

Se il file da caricare è un accessorio da scrivania, il registro r10L deve specificare se la parte di schermo nascosta dev'essere salvata, in modo che al termine dell'esecuzione GEOS possa ripristinare la situazione precedente all'attivazione del desk accessory. Alcune applicazioni non usano il buffer di schermo per mantenere una copia dello schermo originale e quindi, dopo l'esecuzione dell'accessorio da scrivania, devono ripristinare lo schermo originale utilizzando metodi propri.

FindFile

Funzione: Cerca e carica in memoria il File Entry del file specificato.

Indirizzo: $C20B

Chiamata da: GetFile, FastDelFile, RenameFile

Parametri: r6 — puntatore alla stringa che contiene il nome del file. La stringa dev'essere a terminazione nulla e non più lunga di 16 caratteri, terminatore escluso

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Accede a: disco — i File Entry nella directory del disco

Restituisce:

x	codice d'errore da disco
diskBlkBuf	il blocco della directory che contiene il File Entry cercato
dirEntryBuf	File Entry del file
curDirHead	il Directory Header Block del disco
r1L, r1H	indirizzo T/S del blocco della directory interessato
r5	puntatore al File Entry del file memorizzato all'interno del buffer diskBlkBuf

Distrugge: a, y, r4, r6

Sinossi: FindFile richiede soltanto il nome del file da 16 caratteri e la presenza di un disco, nel drive correntemente attivato, dove effettuare la ricerca del file. Il disco deve già essere stato "aperto" tramite OpenDisk o NewDisk (le quali a loro volta richiedono che sia già stata eseguita la routine SetDevice). La routine cerca il File Entry del file scorrendo i blocchi della directory, compreso il blocco Off Page se il disco è in formato GEOS. Se lo trova, il blocco della directory in cui si trova e il File Entry stesso vengono trasferiti in memoria per essere utilizzati con altre routine di lettura e scrittura su disco. Se il file non si trova sul disco o si verifica un errore nell'accesso al disco, per operare le appropriate scelte l'applicazione può consultare il registro x, che contiene il codice dell'errore. Normalmente il nome del file specificato per la ricerca è uno di quelli elencati dalla routine FindFTypes o una stringa in input dall'utente.

SaveFile

Funzione: Salva un'area di memoria su disco come un file a struttura SEQUENTIAL. SaveFile viene anche impiegata per creare su disco un file vuoto a struttura VLIR.

Indirizzo: $C1ED

Chiama: GetDirHead, SetGDirEntry, PutDirHead

Parametri:

r9	puntatore al blocco File Header. Quando il File Header viene memorizzato su disco, i primi due byte contengono rispettivamente i valori $00 e $FF. Quando il blocco viene passato a SaveFile, questi due byte devono contenere un puntatore che individua in memoria una stringa a terminazione nulla contenente il nome del file. Per maggiori dettagli sulla struttura del blocco File Header si consulti il capitolo 13
r10L	numero della pagina (blocco) della directory nella quale la routine deve provare a memorizzare il File Entry del file da salvare su disco. La suddivisione in pagine della directory è utile per raffigurare i fogli del bloc notes utilizzato da deskTop per indicare i file in directory. Ognuna di queste pagine è memorizzata in un blocco della traccia 18, riservata alla directory del disco. Se per esempio l'applicazione passa il numero 4 in r10L, SaveFile tenta di memorizzare il File Entry del file nel quarto blocco della directory (si assume che il primo blocco, quello che contiene la BAM, sia il blocco 0)

Restituisce:

fileHeader	contiene il blocco File Header come è stato memorizzato su disco
curDirHead	contiene il Directory Header Block ottenuto dalla chiamata a GetDirHead
r6	puntatore alla tavola fileTrScTab
r9	puntatore al blocco File Header in memoria
dirEntryBuf	contiene il nuovo File Entry per il file
disco	il disco contiene la nuova BAM e la nuova Directory Header
disco	il nuovo blocco File Header e il file memorizzati su disco
disco	se il file è a struttura VLIR viene creata la tavola indice
x	codice d'errore da disco

Distrugge: a, y, r0 - r8

Sinossi: SaveFile salva su disco un'area di memoria per un file a struttura SEQUENTIAL compatibile con lo standard GEOS. Per effettuare questa operazione, SaveFile ha bisogno che siano passati il blocco File Header del file e il numero della pagina della directory nella quale memorizzare il File Entry. Il File Header contiene la maggior parte delle informazioni necessarie a SaveFile per creare il File Entry in directory e memorizzare il file su disco: l'icona, il tipo di file, l'indirizzo di caricamento, di esecuzione e di fine file, la stringa della versione. Per maggiori dettagli sul blocco File Header consultare il capitolo 13.

L'unica informazione che non appare nel blocco File Header, ma dev'essere presente nel File Entry del file salvato su disco è il nome. SaveFile si aspetta quindi di trovare nei primi due byte del blocco File Header, residente ancora in memoria, l'indirizzo della stringa a terminazione nulla che contiene il nome da dare al file. I nomi di file possono arrivare sino a 16 caratteri significativi più uno 0 come terminatore.

SaveFile può essere utilizzata per creare un file a struttura VLIR, vuoto. L'unica differenza, in questo caso, consiste negli indirizzi di inizio file (indirizzo di caricamento) e di fine file che devono essere rispettivamente $0000 e $FFFF. In questo modo la routine sa che non deve allocare alcun blocco di dati su disco. La struttura GEOS del file memorizzata nel File Header deve indicare che il file è organizzato con la struttura VLIR. Quando SaveFile chiama SetGDirEntry, viene allocato su disco un blocco per contenere la tavola indice del file. In questo blocco SaveFile memorizza una tavola indice vuota.

Ogni pagina del bloc notes raffigurato in deskTop rappresenta un blocco della directory allocato nella traccia 18. Il numero passato in r10L è il numero della pagina nella quale SaveFile prova a memorizzare il File Entry del file (si noti che la terza pagina, o blocco, della directory non è memorizzata nel terzo settore della traccia 18, ma nel settore che rende minima la distanza con il blocco precedente). Se la pagina richiesta risulta piena, SaveFile esamina le pagine che seguono fino a quando non trova uno spazio libero per memorizzare il File Entry. Al limite, la routine è in grado di allocare un nuovo settore come blocco della directory, se non ha trovato spazio nei blocchi esaminati.

DeleteFile

Funzione: Cancella un file C-64, GEOS SEQUENTIAL o VLIR dal disco.

Indirizzo: $C238

Chiama: FindFile, FreeFile

Parametri: r0 puntatore al nome del file (stringa a terminazione nulla)

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce:

x	codice d'errore da disco
curDirHead	il Directory Header Block in memoria viene aggiornato per indicare i cambiamenti effettuati su disco
disco	la BAM viene aggiornata
disco	viene aggiornato il blocco della directory che conteneva il File Entry
disco	il blocco File Header e la tavola indice (se file VLIR) vengono cancellati dal disco come il resto del file
dirEntryBuf	restituito per via della chiamata a FindFile
r9	punta al buffer dirEntryBuf

Distrugge: a, y, r0 - r9

Sinossi: Cancella dal disco il file con il nome specificato. Per disallocare i blocchi del file, la routine utilizza esclusivamente le routine del turbo di GEOS, e non accede ad alcuna routine del DOS. Tramite DeleteFile possono essere

cancellati dal disco file di qualunque dimensione e anche con struttura VLIR. Il File Entry del file viene rimosso dalla directory, ma ne resta una copia nel buffer dirEntryBuf. Ogni settore del file è contrassegnato nella BAM come disallocato. Se il file è di tipo C-64 REL, il puntatore al side-sector non è 0, e punta a una concatenazione di blocchi. Anche in questo caso, i blocchi vengono cancellati. Se il file è a struttura VLIR vengono cancellati la tavola indice, il File Header e tutti i record di cui si compone.

RenameFile

Funzione: Assegna al file un nuovo nome.

Indirizzo: $C259

Chiama: FindFile

Parametri:

r6	puntatore al vecchio nome (a terminazione nulla)
r0	puntatore al nuovo nome (a terminazione nulla)

Restituisce:

dirEntryBuf	contiene il File Entry del file aggiornato con il nuovo nome
diskBlkBuf	conserva una copia del blocco della directory che contiene il File Entry del file. La stringa del vecchio nome viene sostituita all'interno del File Entry con la stringa del nome nuovo e il buffer diskBlkBuf viene nuovamente memorizzato sul disco
x	codice d'errore da disco

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Distrugge: a, x, y, r1, r4 - r6

Sinossi: Questa routine assegna al file specificato un nuovo nome. FindFile carica in memoria il File Entry del file al quale si desidera cambiare il nome. Subito dopo, la routine aggiorna il File Entry con il nuovo nome e lo riscrive sul disco. Il nome puntato da r6 (il vecchio nome) non dev'essere memorizzato nel buffer dskBlkBuf dal momento che quella zona di memoria viene cancellata quando FindFile scorre i blocchi della directory alla ricerca del file.

EnterDeskTop

Funzione: Inizializza il sistema a uno stato di default, carica in memoria deskTop e la esegue.

Indirizzo: $C22C

Chiamata da: Tutte le applicazioni quando concludono la loro esecuzione

Parametri: Nessuno

Restituisce: Il sistema inizializzato a uno stato di default tramite una partenza "a caldo"

Distrugge: Il sistema è interamente inizializzato

Sinossi: Quando un'applicazione termina la propria esecuzione, dovrebbe eseguire l'istruzione jmp EnterDeskTop. Questa routine effettua una partenza a caldo del sistema, carica in memoria deskTop e la esegue. Se GEOS non trova sul disco corrente l'applicazione deskTop, visualizza un box di dialogo e chiede che venga inserito un disco contenente deskTop.

CalcBlksFree

Funzione: Calcola il numero di blocchi liberi sul disco scorrendo la BAM.

Indirizzo: $C1DB

Chiamata da: BlkAlloc, NxtBlkAlloc, SetGEOSDisk

Parametri: r5 — punta al buffer che contiene il Directory Header Block. Normalmente questo buffer è curDirHead

Restituisce:
r4 — il numero di blocchi liberi presenti su disco (è una word)
r5 — inalterato

Distrugge: a, y

Sinossi: Dopo aver ricevuto il Directory Header Block del disco corrente, CalcBlksFree accede alla BAM e calcola il numero di blocchi liberi del disco. Sul disco vi sono 35 tracce. All'interno della BAM vi è un gruppo di bit per ogni traccia. Ognuno di questi gruppi è composto da quattro byte. Il primo byte di ogni gruppo indica quanti blocchi sono contenuti nella traccia. I rimanenti tre byte contengono un bit per ogni settore della traccia. La BAM è allocata dal byte in posizione $04 al byte in posizione $8F (dal 4 al 143) del Directory Header Block, traccia 18 settore 0.

15 ROUTINE DI LIVELLO INTERMEDIO

Nel capitolo precedente abbiamo trattato le routine d'alto livello di cui dispone GEOS per gestire il sistema dei file. Questo capitolo descrive le routine di livello intermedio. Combinandole opportunamente fra loro si possono realizzare alcune funzioni particolari che solo per ragioni di spazio non sono state inserite nel Kernel di GEOS, come per esempio le routine di copia dei file e dei dischi. Queste funzioni non direttamente previste da GEOS sono facili da realizzare e possono essere ottimizzate per le specifiche applicazioni. Nel livello intermedio sono incluse le seguenti routine:

GetBlock	GetFreeDirBlk
PutBlock	BlkAlloc
GetFHdrInfo	NxtBlkAlloc
ReadFile	SetNextFree
WriteFile	FindBAMBit
ReadByte	FreeBlock
GetDirHead	SetGDirEntry
PutDirHead	BldGDirEntry
NewDisk	FollowChain
LdApplic	FastDelFile
LdFile	FreeFile
LdDeskAcc	ChangeDiskDevice
RstrAppl	StartAppl

Prima d'iniziare la descrizione di queste routine, ricordiamo che tutte restituiscono nel registro x il codice d'errore da disco. Questo codice informa sull'esito del tentativo d'accesso al disco (positivo se $x = 0$, negativo se $x \neq 0$). Nel caso che x sia diverso da zero, il registro contiene il codice che identifica l'errore. Per un riassunto di tutti gli errori da disco gestiti da GEOS consultare l'appendice.

GetBlock

Funzione: Routine di livello intermedio standard per prelevare un blocco da disco.

Indirizzo: $C1E4

Chiamata da: GetFHdrInfo, UpdateRecordFile

Chiama: EnterTurbo, InitForIO, ReadBlock, DoneWithIO

Parametri:

r1L, r1H	indirizzo T/S del blocco da leggere
r4	puntatore al buffer nel quale memorizzare i dati prelevati da disco (generalmente diskBlkBuf)

Preparazione del drive: Il drive dev'essere inizializzato attraverso OpenDisk o NewDisk, e deve corrispondere al dispositivo selezionato sul bus seriale attraverso SetDevice.

Restituisce:

x	codice d'errore da disco
r4	puntatore al buffer in memoria nel quale è stato memorizzato il blocco prelevato da disco (è rimasto inalterato)
r1	inalterato

Distrugge: a, x, y

Sinossi: Passando l'indirizzo T/S del blocco che dev'essere letto, GetBlock lo preleva dal disco e lo memorizza nel buffer specificato. GetBlock carica un blocco da disco e lo memorizza nella memoria del C-64 utilizzando il codice turbo di cui è dotato GEOS. Dopo essere stato trasferito in memoria, il blocco contiene ancora nei primi due byte l'indirizzo T/S del settore successivo, se appartiene a un concatenamento. GetBlock è una routine di livello intermedio e si presta a essere impiegata per trasferire un singolo blocco dal disco alla memoria del C-64 (generalmente nel buffer diskBlkBuf), o per realizzare funzioni personalizzate. Questo secondo tipo d'impiego è poco indicato per i programmatori non molto esperti.

Ecco uno schema a blocchi che illustra le principali operazioni compiute da GetBlock:

disabilita gli interrupt,
  disabilita gli sprite,
    attiva il codice turbo nel drive (se necessario lo trasferisce nel drive),
    legge il blocco,
      disattiva il codice turbo nel drive,
    attiva gli sprite, e
  attiva gli interrupt.

GetBlock trasferisce sempre e solo i 256 byte che compongono il blocco indicato, anche se si tratta dell'ultimo blocco di un settore. Fino alla versione 1.2 del Kernel, nel caso che il primo byte del blocco fosse 0 (ultimo blocco del concatenamento), GetBlock trasferiva solo i byte significativi che conteneva. Dalla versione 1.3 in avanti, la routine GetBlock legge in ogni caso tutti i byte del settore indicato, significativi e non. Questo cambiamento è stato effettuato per permettere alle applicazioni di utilizzare GetBlock anche nella lettura di blocchi non organizzati secondo il consueto concatenamento Traccia/Settore. Questo caso interessa particolarmente alcuni dischi disponibili per il C-64, contenenti dati grafici memorizzati secondo strutture inconsuete.

PutBlock

Funzione: Memorizza su disco un blocco di memoria. Se il blocco appartiene a un concatenamento deve contenere l'indirizzo T/S del successivo.

Indirizzo: $C1E7

Chiama: EnterTurbo, InitForIO, WriteBlock, DoneWithIO

Parametri:

r1L, r1H	indirizzo T/S del settore su disco nel quale viene memorizzato il blocco
r4	puntatore in memoria all'inizio del blocco, di solito diskBlkBuf

Preparazione del drive: Il drive dev'essere inizializzato attraverso OpenDisk o NewDisk, e deve corrispondere al dispositivo selezionato sul bus seriale attraverso SetDevice.

Restituisce:

x	codice d'errore da disco
r4	inalterato
r1	inalterato

Distrugge: a, x, y

Sinossi: PutBlock trasferisce un blocco dalla memoria del C-64 al disco corrente utilizzando il codice turbo. Perché un file sia trasferito su disco correttamente, i blocchi che lo compongono devono essere concatenati fra loro. Questo significa che i primi due byte di ogni settore ancora residente in memoria devono contenere l'indirizzo T/S del blocco successivo. Una volta preparato il blocco, mandando in esecuzione PutBlock si trasferiscono i dati nel settore specificato.

Di solito il blocco da trasferire è memorizzato nel buffer diskBlkBuf. PutBlock viene generalmente utilizzato all'interno di un loop che individua il successivo settore libero su disco, aggiorna il puntatore T/S nel blocco ancora in memoria e chiama PutBlock per memorizzarlo nel settore correntemente libero (nota bene: il settore correntemente libero non è il successivo settore libero).

Di solito si ricorre a PutBlock quando l'applicazione fa alcune variazioni nei codici contenuti in un blocco del file prelevato tramite GetBlock, e si deve riscrivere il blocco aggiornato su disco. Se si rende necessario aggiungere

uno o più blocchi a un file preesistente, l'applicazione deve utilizzare la routine NxtBlkAlloc, che fa riferimento alla BAM del disco, per allocare gli opportuni settori. L'indirizzo T/S del blocco successivo dev'essere memorizzato nella prima word di ogni blocco. PutBlock è una routine di livello intermedio e la sua utilità è probabilmente limitata alle applicazioni che vogliono realizzare funzioni di accesso al disco personalizzate. Normalmente le routine d'alto livello sono sufficienti a soddisfare quasi tutte le esigenze di questo tipo.

Ecco un diagramma a blocchi che illustra le principali operazioni compiute da PutBlock:

```
disabilita gli interrupt,
   disabilita gli sprite,
      attiva il codice turbo nel drive (se necessario lo trasferisce nel drive),
      scrive il blocco,
         disattiva il codice turbo nel drive,
      attiva gli sprite, e
   attiva gli interrupt.
```

Il codice turbo residente nel drive mantiene il possesso del bus seriale, in modo che nessun altro dispositivo esterno possa accedere al bus mentre è in atto una comunicazione con il drive.

GetFHdrInfo

Funzione: Carica il blocco File Header del file e lo memorizza nel buffer fileHeader.

Indirizzo: $C229

Chiamata da: LdFile, LdDeskAcc

Chiama: GetBlock

Parametri:

r9	puntatore in memoria al File Entry del file, generalmente memorizzato nel buffer dirEntryBuf
dirEntryBuf	buffer che contiene il File Entry. Sebbene GetFHdrInfo non abbia bisogno dell'intero blocco nel quale è contenuto il File Entry interessato, questo viene ugualmente caricato in memoria dalla routine FindFile che GetFHdrInfo utilizza per ricercare il File Entry del file, e viene memorizzato nel buffer diskBlkBuf

Restituisce:

r1	indirizzo T/S del primo blocco di dati del file ottenuto dal File Entry associato. Se il file è a struttura VLIR, questo primo blocco contiene la tavola indice dei record
r7	indirizzo di caricamento del file, prelevato dal blocco File Header
fileHeader	buffer contenente il blocco File Header
fileTrScTab	i primi due byte contengono l'indirizzo T/S del blocco File Header su disco
x	codice d'errore da disco

Distrugge: a, y, r4

Sinossi: GetFHdrInfo legge l'indirizzo T/S del blocco File Header prelevato dal File Entry e lo memorizza nel buffer fileHeader. Aggiorna il registro r1 con l'indirizzo T/S del primo blocco del file, e restituisce in r7 l'indirizzo di caricamento del file.

ReadFile

Funzione: Routine di livello intermedio utilizzata per leggere i dati contenuti in una concatenazione di blocchi su disco. I primi due byte di ogni blocco, cioè l'indirizzo T/S del blocco successivo, non vengono ovviamente presi in considerazione durante la lettura dei dati.

Indirizzo: $C1FF

Chiamata da: LdFile, LdDeskAcc

Chiama: EnterTurbo, InitForIO, ReadBlock, DoneWithIO

Parametri:

r7	Indirizzo a cui iniziare la memorizzazione dei dati
r1L, r1H	Indirizzo T/S del primo blocco del concatenamento
r2	dimensione del buffer destinazione espressa in byte

Restituisce:

fileTrScTab	lista degli indirizzi T/S dei blocchi del file o del record. Un file può essere composto al massimo da 127 blocchi (32.258 byte). GetFHdrInfo memorizza nella prima word della tavola fileTrSc-Tab l'indirizzo T/S del blocco File Header del file. Quando si utilizza GetFile per caricare un file in memoria, viene eseguita la routine ReadFile per completare la tavola fileTrScTab
x	codice d'errore da disco: se il file è composto da un numero di blocchi che eccede le dimensioni del buffer (contenute in r2), GEOS restituisce in x l'errore BFR_OVERFLOW per indicare che sono stati oltrepassati i limiti di capienza del buffer. Se un blocco non può essere memorizzato interamente nel buffer, non viene letto. Se non si verifica nessun errore, GEOS restituisce 0 in x
r7	punta in memoria la locazione che segue l'ultimo byte memorizzato
r5L	offset all'ultima word significativa memorizzata nella tavola fileTrScTab. Questa contiene l'indirizzo T/S dell'ultimo blocco letto da disco
r1	nel caso che i limiti di capienza del buffer siano stati oltrepassati, e si verifichi quindi l'errore BFR_OVERFLOW, r1 contiene l'indirizzo T/S del blocco che non è stato letto

Distrugge: a, y, r1 - r4

Sinossi: ReadFile legge i dati contenuti in un file e li trasferisce in memoria. Mentre compie questa operazione, aggiorna anche la tavola fileTrScTab con gli indirizzi T/S dei blocchi letti. ReadFile viene mandata in esecuzione in seguito alla chiamata da parte dell'applicazione di una delle routine di caricamento d'alto livello, come GetFile. Per esempio, GetFile chiama LdFile che a sua volta chiama ReadFile per leggere un normale file a struttura SEQUENTIAL o anche a struttura VLIR. Nel caso di un file a struttura VLIR, r1 deve contenere l'indirizzo T/S del primo record. Prima di eseguire ReadFile, LdFile chiama GetFHdrInfo per leggere il File Header e memorizzarlo nel buffer fileHeader. Accedendo al File Header, LdFile preleva l'indirizzo T/S del blocco indice e carica in memoria la tavola indice del file a struttura VLIR. Legge quindi dalla tavola l'indirizzo T/S del record desiderato e lo trasmette in r1 a ReadFile, che a sua volta memorizza il contenuto di r1 nel buffer fileTrScTab alla locazione fileTrScTab+2. fileTrScTab+2 individua la prima word libera dopo quella aggiornata da GetFHdrInfo, per memorizzarvi l'indirizzo T/S del primo record.

WriteFile

Funzione: Trasferisce su disco una particolare area di memoria. I dati vengono salvati in una concatenazione di settori liberi già allocati per questa operazione. La routine verifica il corretto trasferimento di tutti i blocchi.

Indirizzo: $C1F9

Chiamata da: SaveFile

Chiama: Il codice turbo di accesso al disco

Parametri:

r7	puntatore all'inizio dell'area di memoria da trasferire su disco
r6	puntatore alla tavola di concatenamento dei blocchi allocati su disco per contenere il file (fileTrScTab)

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce:

x	codice d'errore da disco
fileTrScTab	la tavola degli indirizzi dei blocchi allocati su disco, inalterata

Distrugge: a, y, r1, r2, r4, r6, r7

Sinossi: WriteFile trasferisce su disco un'area di memoria specificata. Si tratta di una routine di livello molto basso ed è necessario che l'applicazione, prima di eseguirla, abbia aggiornato diverse variabili: il File Entry dev'essere già stato allocato dalle routine SetGDirEntry, BldGDirEntry e/o GetFreeDirBlk, dev'essere già stata preparata la tavola fileTrScTab degli indirizzi T/S dei blocchi

che la routine può utilizzare per allocare il file, e i blocchi in essa indicati devono essere già stati allocati tramite la routine BlkAlloc. Ovviamente per compiere queste operazioni dev'essere già stato creato anche il blocco File Header. Oltre alla preparazione delle variabili significative, la routine WriteFile richiede solo che venga specificato il contenuto della tavola fileTrScTab e dei registri r6 e r7.

WriteFile trasferisce su disco i dati prelevati in memoria a partire dall'indirizzo indicato in r7. Ogni volta che la routine preleva un blocco di 254 byte dalla memoria, accede alla tavola fileTrScTab per leggere l'indirizzo T/S del successivo blocco allocato su disco. WriteFile crea il blocco da trasferire aggiungendo ai due byte che contengono l'indirizzo T/S i 254 byte che legge dalla memoria. Il blocco da 256 byte così ottenuto, viene infine memorizzato su disco. La traccia e il settore del blocco successivo vengono prelevati dal buffer fileTrScTab. WriteFile individua la fine dell'area di memoria da trasferire, scorrendo gli indirizzi T/S dei blocchi lungo la tavola fileTrScTab. Quando incontra un indirizzo T/S il cui primo byte è 0, WriteFile calcola (tramite il valore del secondo byte) qual è l'ultima locazione di memoria da trasferire. Infatti, un blocco che ha come primo byte 0 dev'essere l'ultimo della concatenazione e il secondo byte deve contenere il numero di byte significativi all'interno del blocco.

ReadByte

Funzione: Permette di prelevare, a un ritmo di un byte alla volta, i dati memorizzati in una concatenazione di blocchi su disco. I registri r1, r4, r5 e il buffer allocato per l'operazione non devono essere alterati durante la sequenza di chiamate a questa routine, finché non sono stati prelevati tutti i dati.

Indirizzo: $C2B6

Chiama: GetBlock

Parametri: Solo alla prima chiamata:

r1	indirizzo T/S del primo blocco da leggere
r4	puntatore a un buffer da 256 byte, di solito diskBlkBuf
r5	0, indice al prossimo byte da leggere inizializzato a 0 per la prima chiamata

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

r1	indirizzo T/S del successivo blocco da leggere
r4	inalterato
r5	incrementato, indice al successivo byte da leggere
x	codice d'errore da disco ("nessun errore" restituisce x = 0) Quando si verifica l'errore BFR_OVERFLOW, significa che la routine ha restituito l'ultimo byte del concatenamento di blocchi
a	il byte dato prelevato dal file
Z	il flag Z nel PSW è impostato a 1 se il registro x è diverso da 0; in questo modo si rende più rapido il controllo sull'eventuale presenza di un errore

Distrugge: y

Sinossi: ReadByte permette all'applicazione di ricevere uno alla volta i byte contenuti in una concatenazione di blocchi su disco. In realtà l'input dei dati da disco è ancora a blocchi, ma questi vengono memorizzati uno alla volta in un buffer. ReadByte, richiamata periodicamente dall'applicazione, accede al buffer e lo svuota, caricando quindi il blocco successivo. Sia i file a struttura SEQUENTIAL sia quelli a struttura VLIR possono essere letti attraverso questa routine. L'applicazione deve specificare l'indirizzo T/S del primo blocco da leggere e chiamare ReadByte per ottenere il primo byte. Per ottenere il successivo, l'applicazione deve chiamare nuovamente ReadByte e così via. Quando si verifica l'errore BFR_OVERFLOW, significa che la routine ha cercato di leggere un byte dopo la fine del file, e quindi che la precedente chiamata alla routine aveva restituito l'ultimo byte del concatenamento.

GetDirHead

Funzione: Preleva la Directory Header e la BAM del disco, contenute nel Directory Header Block, e le trasferisce in memoria. In pratica la routine trasferisce il contenuto del blocco collocato a traccia 18 e settore 0 della directory.

Indirizzo: $C247

Chiamata da: OpenDisk, SetGEOSDisk, BlkAlloc, FreeFile, FastDelFile

Chiama: GetBlock

Parametri: curDrive — questa variabile globale contiene il valore 8 o 9 per indicare il drive correntemente attivo. Il drive indicato dev'essere il dispositivo che utilizza in quel momento il bus seriale, ovvero quello selezionato tramite SetDevice, e dev'essere inizializzato tramite una chiamata a NewDisk o a OpenDisk (OpenDisk chiama a sua volta NewDisk)

Restituisce: curDirHead — questo buffer contiene il Directory Header Block del disco corrente

Distrugge: a, x, y, r1, r4

Sinossi: GetDirHead legge dal disco corrente il Directory Header Block, che comprende anche la BAM, e lo memorizza a curDirHead. La routine SetDevice dev'essere già stata chiamata per aggiornare curDrive con il numero del drive che si vuole attivare cedendogli il bus seriale. Inoltre dev'essere già stata eseguita la routine NewDisk (oppure OpenDisk), cioè il drive deve già essere stato preparato per accedere al disco

Per questa routine sono previsti alcuni miglioramenti che dovrebbero renderla compatibile con qualsiasi tipo di drive, come per esempio la capacità di accedere a BAM di diverso formato. GEOS conserva una copia della BAM in memoria mentre alloca e disalloca blocchi su disco. I cambiamenti di stato dei blocchi sul disco sono riportati nella copia della BAM in memoria. Quando sono state operate le dovute variazioni, la copia aggiornata della BAM dev'essere riallocata sul disco tramite la routine PutDirHead. È per questo che l'utente non dovrebbe mai rimuovere il disco dal drive durante gli accessi. Se la nuova BAM non viene riallocata, il disco è da considerare parzialmente rovinato, anche se non in maniera definitiva.

PutDirHead

Funzione: Rialloca su disco la copia del Directory Header Block conservata in memoria.

Indirizzo: $C24A

Chiamata da: SaveFile, FreeFile, FastDelFile, SetGEOSDisk

Chiama: PutBlock

Parametri:

curDirHead	una copia valida del Directory Header Block per il disco corrente
curDrive	questa variabile globale contiene il valore 8 o 9 per indicare il drive correntemente attivo. Il drive indicato dev'essere il dispositivo che utilizza in quel momento il bus seriale, ovvero quello selezionato tramite SetDevice, e dev'essere inizializzato tramite una chiamata a NewDisk o a OpenDisk (OpenDisk chiama a sua volta NewDisk)

Restituisce:

disco	il Directory Header Block su disco aggiornato con la copia memorizzata a curDirHead

Distrugge: a, x, y, r0 - r15

Sinossi: Scrive su disco il Directory Header Block correntemente memorizzato nel buffer curDirHead. PutDirHead, in seguito, sarà in grado di trasferire su disco tutte le porzioni delle BAM di drive di diverso tipo.

NewDisk

Funzione: Trasferisce la BAM del disco corrente nella memoria interna del drive.

Indirizzo: $C1E1

Chiamata da: OpenDisk

Chiama: EnterTurbo, InitForIO, DoneWithIO

Parametri: curDrive — deve contenere il numero del drive corrente

Restituisce:
drive — nella memoria del drive è residente la BAM del disco corrente
x — codice d'errore da disco

Distrugge: a, y, r1

Sinossi: Il disk drive 1541 memorizza il byte d'identificazione ID del disco corrente nella sua memoria interna. Quando l'utente inserisce un nuovo disco nel drive, il nuovo byte ID dev'essere memorizzato dallo stesso drive prima di qualunque altra operazione. NewDisk ordina al drive di leggere la BAM del disco appena inserito. Il drive dovrebbe già essere in ascolto sul bus seriale. Subito dopo SetDevice, che mette in condizione "di ascolto" il drive, generalmente viene chiamata NewDisk.

Se il codice turbo all'interno del drive non è in esecuzione, la routine lo installa e lo lascia in esecuzione. NewDisk è una subroutine di OpenDisk.

LdApplic - Carica l'applicazione

Funzione: Carica ed esegue un'applicazione GEOS compatibile.

Indirizzo: $C21D

Chiamata da: GetFile

Chiama: LdFile

Parametri:

r9 — puntatore al File Entry del file (generalmente memorizzato nel buffer dirEntryBuf)

r0L — flag loadOpt
- bit 0
  - 0 segue la procedura standard di caricamento
  - 1 carica il file all'indirizzo specificato in r7 (la costante per questo bit è ST_LD_AT_ADDR)
- bit 6 (solo per file eseguibili)
  - 0 il file non è stato selezionato per essere stampato direttamente
  - 1 deskTop, per esempio, imposta a 1 questo bit quando l'utente muove l'icona di un file dati sopra l'icona della stampante e preme il pulsante del mouse. L'applicazione stampa il file e conclude la sua esecuzione (la costante per questo bit è ST_PR_DATA)
- bit 7 (solo per file eseguibili)
  - 0 non è specificato alcun file dati
  - 1 l'applicazione da caricare è quella che ha creato il file dati selezionato dall'utente per essere subito elaborato (la costante per questo bit è ST_LD_DATA)

r2 e r3 — vengono passati solo se:
1) dev'essere caricata un'applicazione
2) il bit 6 o il bit 7 in r0L (o entrambi) sono impostati a 1

r2 — puntatore al nome del disco che contiene il file dati che l'applicazione deve elaborare automaticamente. Il registro r2 e il contenuto del buffer dataDiskName vengono passati all'applicazione. Questa, una volta che è stata caricata e

mandata in esecuzione, elabora il file dati come specificato nel bit 6 di loadOpt

r3 punta al buffer che contiene il nome del file dati. L'applicazione utilizza questo puntatore per caricare il file dati in memoria ed elaborarlo automaticamente. r3 e il contenuto del buffer dataFileName vengono passati all'applicazione. Questa, una volta che è stata caricata in memoria e mandata in esecuzione, elabora il file dati come specificato nel bit 6 di loadOpt

r7 contiene l'indirizzo al quale il file dev'essere memorizzato in memoria se il bit 0 in r0L, individuato da ST_LD_AT_ADDR, è impostato a 1

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine d'alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce: Restituisce l'applicazione in memoria. Se il file è a struttura VLIR la routine carica in memoria il primo record. Se il flag loadOpt non specifica la richiesta di un rilocamento del file a una particolare zona di memoria, LdApplic inizializza il sistema a uno stato di default ed esegue l'applicazione. La routine in questo caso non restituisce il controllo al codice che l'ha chiamata. Quando invece loadOpt richiede esplicitamente un rilocamento del file a un indirizzo diverso da quello indicato nel blocco File Header, LdApplic, dopo aver caricato il file in memoria all'indirizzo indicato da r7, restituisce il controllo al codice che l'ha chiamata, perché non avrebbe senso mandare in esecuzione un'applicazione che è stata caricata a un indirizzo diverso da quello dove normalmente viene collocata. Sarebbe molto improbabile che funzionasse ugualmente

r7 loadAddr, contiene l'indirizzo d'esecuzione (start address) prelevato dal blocco File Header

r2, r3 e r0 — vengono passati all'applicazione o all'accessorio da scrivania

r5L — offset dall'inizio del fileTrScTab all'ultima word significativa per il file

dataDiskName inalterato
dataFileName inalterato
dirEntryBuf inalterato

x — codice d'errore da disco: se il file è composto da più di 127 blocchi, si verifica l'errore BFR_OVERFLOW e non viene letto il blocco 128. La presenza di un qualsiasi errore da disco costringe la routine LdApplic a restituire il controllo al codice che l'ha chiamata

r1 — nel caso si verifichi l'errore BFR_OVERFLOW, r1 contiene l'indirizzo T/S del blocco che non è stato letto.
LdApplic effettua una partenza "a caldo" del sistema. Per maggiori dettagli consultare la configurazione di sistema in partenza "a caldo" illustrata nel capitolo 20

Distrugge: Tutti gli pseudoregistri che non sono citati tra i parametri restituiti. Se non è specificato alcun rilocamento forzato, e se non si verificano errori, la routine LdApplic non restituisce il controllo al codice che l'ha chiamata

Sinossi: LdApplic viene chiamata da GetFile per caricare ed eseguire un'applicazione GEOS compatibile. Prima di chiamare LdApplic, GetFile chiama la routine FindFile. Quindi, normalmente, LdApplic dipende dalle variabili impostate da FindFile. Ci sono tre motivi perché un'applicazione debba essere caricata in memoria:

1) l'utente ha selezionato l'applicazione da deskTop, e questa dev'essere eseguita come di consueto
2) l'utente ha selezionato da deskTop un file dati, e l'applicazione dev'essere caricata ed eseguita per elaborarlo automaticamente
3) l'applicazione dev'essere eseguita per stampare automaticamente un file dati e subito dopo deve restituire il controllo a deskTop.

Per indicare quale tra questi è il motivo per cui l'applicazione è stata caricata ed eseguita, viene passato a LdApplic il flag loadOpt. Se

l'applicazione è stata eseguita in seguito alla selezione di un file dati o alla richiesta di stampa di un file dati, la routine che esegue la chiamata (di solito deskTop) deve passare il nome del file dati e del disco sul quale è residente. Poi l'applicazione deve leggere il contenuto del flag loadOpt e operare di conseguenza.

Grazie a LdApplic è possibile anche caricare un file in memoria a un indirizzo diverso da quello consueto. Questa opzione serve più che altro per i file dati, dal momento che è improbabile che un'applicazione venga realizzata con codici rilocabili e quindi che sia eseguibile in qualunque area della memoria. Se LdApplic si trova a caricare un file a un indirizzo diverso da quello specificato nel blocco File Header, a operazione ultimata restituisce il controllo al codice di chiamata. Sarebbe del resto assurdo mandare in esecuzione un'applicazione rilocata a un indirizzo arbitrario. Si deve inoltre evitare che l'applicazione caricata, e rilocata a un diverso indirizzo, vada a sovrapporsi al codice di chiamata, perché altrimenti LdApplic si troverebbe a restituire il controllo a un codice diverso da quello di chiamata.

LdFile - Carica il file

Funzione: Carica un file in memoria.

Indirizzo: $C211

Chiamata da: GetFile, LdApplic

Chiama: GetFHdrInfo, ReadFile

Parametri:

r9 puntatore al File Entry, memorizzato nel buffer dirEntryBuf

r0L flag loadOpt per la selezione delle opzioni di caricamento
bit 0
- 0 segue le informazioni di caricamento memorizzate nel blocco File Header
- 1 carica il file all'indirizzo specificato in loadAddr (r7). La costante per questo bit è ST_LD_AT_ADDR

r7 loadAddr: parametro opzionale. Se il bit ST_LD_AT_ADDR è impostato a 1 in loadOpt, loadAddr dovrebbe contenere l'indirizzo di caricamento per il file

Preparazione del drive

Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce:

Il file	caricato in memoria. Il controllo viene restituito al codice di chiamata
loadOpt	il flag per le opzioni di caricamento resta inalterato
loadAddr	l'indirizzo di caricamento alternativo resta inalterato
fileTrScTab+2	lista degli indirizzi T/S dei blocchi che compongono il file o

	il record. Il numero massimo di blocchi che possono comporre il file, o il record, è 127 (32.258 byte). GetFHdrInfo memorizza nella prima word del buffer fileTrScTab l'indirizzo T/S del blocco File Header del file. Di conseguenza, quando viene impiegata GetFile per caricare un file, dev'essere eseguita anche ReadFile per completare la tavola fileTrScTab. Se il file è a struttura VLIR, le word che nel buffer seguono la prima contengono gli indirizzi T/S dei blocchi che compongono il primo record
x	codice d'errore da disco: se si verifica l'errore BFR_OVERFLOW, il file è troppo lungo. Il blocco che oltrepassa il limite non viene letto
r7	punta in memoria il byte successivo all'ultimo memorizzato
r5L	offset dall'inizio del buffer fileTrScTab all'ultima word significativa per il file
r1	se si verifica l'errore BFR_OVERFLOW, r1 contiene l'indirizzo T/S del blocco che non è stato letto perché oltrepassava i limiti

Distrugge: a, x, y, r0 - r10

Sinossi: LdFile viene di solito chiamata dai codici interni di GEOS per caricare un file di sistema come deskTop. Diversamente da GetFile, che carica e manda in esecuzione il file, LdFile, a caricamento ultimato, restituisce sempre il controllo al codice che ha effettuato la chiamata. GetFile utilizza LdFile quando deve caricare in memoria un file non eseguibile.

LdFile utilizza il File Entry del file per caricare il blocco File Header e prelevare le informazioni necessarie per caricare il file. Informazioni che sono inutili se il bit 0 del flag loadOpt è impostato a 1. In questo caso, infatti, l'indirizzo di memoria al quale inizia il trasferimento del file dev'essere indicato in loadAddr.

Se il file è a struttura VLIR, viene caricato in memoria solo il primo record (il record 0). Le opzioni di caricamento previste da LdFile sono valide anche in questo caso.

LdDeskAcc - Carica il desk accessory

Funzione: Questa routine manda in esecuzione un desk accessory. Il desk accessory è un'applicazione che viene eseguita in modo completamente indipendente da qualunque ambiente di lavoro. La sua esecuzione, agli occhi dell'applicazione che l'ha mandata in esecuzione, è completamente trasparente e non altera minimamente l'ambiente di lavoro precedente alla chiamata di LdDeskAcc.

Indirizzo: $C217

Parametri:

r9 puntatore al File Entry, memorizzato nel buffer dirEntryBuf

r0L flag loadOpt per la selezione delle opzioni di caricamento
bit 0
- 0 segue le informazioni per il caricamento memorizzate nel blocco File Header
- 1 carica il file all'indirizzo specificato in loadAddr (r7). La costante per questo bit è ST_LD_AT_ADDR

r7 loadAddr: parametro opzionale. Se il bit ST_LD_AT_ADDR è impostato a 1 in loadOpt, loadAddr dovrebbe contenere l'indirizzo di caricamento per il file

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce: Niente

Distrugge: a, x, y, r0 - r10

Sinossi: LdDeskAcc viene chiamata per mandare in esecuzione un accessorio da scrivania (desk accessory). Prima di svolgere questa operazione, LdDeskAcc salva tutte le variabili globali che caratterizzano l'ambiente di lavoro corrente, comprese le variabili interne di gestione dei menu e delle icone. È un'operazione che preserva l'ambiente di lavoro corrente e permette all'occorrenza di ripristinarlo senza difficoltà. Per sapere con certezza quali aree della memoria vengono salvate durante l'esecuzione di un desk accessory, si consulti la Memory Map riportata in appendice.

Gli accessori da scrivania possono essere allocati in qualunque punto della memoria disponibile alle applicazioni e possono essere anche di dimensioni molto grandi. GEOS, prima di caricare in memoria il desk accessory, provvede a controllare le sue dimensioni e il suo indirizzo di caricamento. Queste informazioni sono necessarie per poter memorizzare su disco, in un file di tipo TEMPORARY (di solito chiamato "swap file"), l'area di memoria che viene sovrascritta dal desk accessory, in modo che alla sua chiusura possa essere correttamente riallocata. È per questo che i desk accessory possono essere eseguiti esclusivamente se il Kernel riesce ad accedere al disco e a memorizzarvi l'area di memoria che viene sovrascritta.

Un accessorio da scrivania dev'essere considerato come un'applicazione, con l'unica limitazione che non può andare a scrivere in aree della memoria non comprese in quella che occupa, e non può chiamare altri desk accessory né aprire box di dialogo. Può però creare qualunque struttura di dati su disco, impiegare menu e icone proprie, come una normale applicazione.

Quando l'accessorio da scrivania conclude la sua esecuzione, non deve far altro che eseguire l'istruzione jmp RstrAppl, in modo che il Kernel provveda a ripristinare interamente la configurazione precedente all'apertura del desk accessory. Il controllo viene restituito al codice che ha richiesto l'esecuzione del desk accessory.

LdDeskAcc, oltre a salvare in una particolare area della memoria di sistema l'ambiente di lavoro corrente, provvede anche a ripristinare con gli appropriati valori di default alcune locazioni di memoria. Nella pagina successiva è riportata una lista delle aree salvate e delle variabili riportate allo stato di default. Ricordiamo che lo stesso vale per i box di dialogo: le aree di memoria salvate prima di aprire un DA sono le stesse che vengono salvate prima di aprire un BD, le variabili riportate a uno stato di default prima dell'apertura di un DA sono le stesse che vengono riportate a uno stato di default prima di aprire un BD.

Variabili globali salvate prima dell'apertura di un desk accessory o di un box di dialogo, e ripristinate alla chiusura:

da currPattern ($0022) a rightMargin ($0037) compresa
da appMain ($849B) a stringY ($84C0) compresa
graphicsMode ($003F)
da spritePointers ($8FF8) a $8FFF compresa
da vicBase ($D000) a msbxpos ($D010) compresa
mobenble ($D015)
da mobprior ($D01B) a mobx2 ($D01D) compresa
da mcmclr0 ($D025) a mcmclr1 ($D026) compresa
da mob1clr ($D027) a mob7clr ($D02E) compresa.

Variabili riportate a uno stato di default prima dell'apertura di un desk accessory o di un box di dialogo:

da currentMode ($002E) a pressFlag ($0039) compresa
da appMain ($849B) a faultData ($84B6) compresa
da spritePointers ($8FF8) a $8FFF compresa.

I valori di default per queste variabili si possono trovare nel capitolo 20.

RstrAppl

Funzione: Questa è la routine che dev'essere eseguita dai desk accessory quando terminano l'esecuzione.

Indirizzo: $C23E

Parametri: Nessuno

Restituisce: L'area di memoria precedentemente salvata su disco, riallocata

Distrugge: In linea di massima, tutti i registri

Sinossi: Questa è la routine che l'accessorio da scrivania deve eseguire, tramite l'istruzione jmp RstrAppl, quando conclude la sua esecuzione. Questa routine provvede a ricaricare da disco l'area di memoria precedentemente salvata, a cancellare il file di tipo TEMPORARY presente su disco e a ripristinare la situazione precedente alla chiamata di LdDeskAcc. Per una lista delle aree di memoria ripristinate da RstrAppl, si consulti la descrizione della routine LdDeskAcc.

GetFreeDirBlk - Alloca lo spazio per un File Entry

Funzione: Indicando una certa pagina della directory, questa routine trova lo spazio libero per memorizzarvi il File Entry di un file. Essa non effettua nessuna variazione sul disco, ma predispone le opportune variabili per la creazione del File Entry. Se la pagina richiesta non ha spazi liberi, la routine cerca uno spazio adatto nelle pagine seguenti.

Indirizzo: $C1F6

Chiamata da: SetGDirEntry

Chiama: GetDiskBlock, AddDirBlock

Parametri:

r10L	numero della pagina della directory all'interno della quale la routine inizia la ricerca di uno spazio libero in grado di contenere un File Entry
curDirHead	contiene il Directory Header Block del disco corrente

Restituisce:

x	codice d'errore da disco
diskBlkBuf	contiene il blocco della directory all'interno del quale la routine ha trovato uno spazio libero
y	contiene un indice, riferito all'inizio del blocco della directory memorizzato nel buffer diskBlkBuf, che punta allo spazio disponibile nella pagina per contenere un File Entry
r10L	contiene il numero della pagina della directory entro la quale la routine ha trovato lo spazio richiesto
curDirHead	contiene il Directory Header Block aggiornato. Attenzione: se la routine ha dovuto allocare un nuovo blocco per la directory, nel caso che le pagine già allocate fossero tutte piene, diventa necessario riscrivere il contenuto del buffer curDirHead su disco, in quanto la routine aggiorna anche la BAM

Distrugge: a, x, y, r0 - r1, r3, r5, r7, r8

Sinossi: Ogni pagina del bloc notes raffigurato in deskTop corrisponde a un blocco della directory (una pagina in deskTop non può contenere più di otto icone e un blocco della directory non può contenere più di otto File Entry). Il valore passato in r10L indica il numero della pagina all'interno della quale la routine

prova a cercare lo spazio per un nuovo File Entry (r10L = 4 indica la pagina quattro del bloc notes di deskTop).

Se non c'è spazio nella pagina richiesta, GetFreeDirBlk estende la ricerca alle pagine successive della directory (i blocchi della directory sono concatenati in sequenza come i file normali). Se necessario, GetFreeDirBlk alloca un nuovo blocco per la directory. Se tutti i 18 blocchi di cui la directory può potenzialmente disporre sono pieni, la routine restituisce un flag d'errore.

Se l'applicazione richiede lo spazio per un File Entry in una pagina non ancora allocata e non contigua a quelle già utilizzate (per esempio, la directory arriva sino a pagina 4 e l'applicazione richiede che la ricerca venga svolta a pagina 10), la routine alloca le necessarie pagine bianche da aggiungere alla directory in modo da rendere disponibile la pagina desiderata.

BlkAlloc

Funzione: Alloca sino a 127 blocchi su disco per creare lo spazio necessario al salvataggio di un file.

Indirizzo: $C1FC

Chiamata da: SaveFile, WriteRecord

Chiama: CalcBlksFree, SetNextFree

Parametri:

r2	numero di byte per i quali la routine deve allocare spazio sul disco. Il numero massimo è 32.258
r6	puntatore all'inizio del buffer da utilizzare per memorizzare la tavola degli indirizzi T/S dei blocchi generata dalla routine. Di solito questo buffer viene allocato a fileTrScTab, o fileTrScTab + 2 se si utilizza la prima word per puntare il nome del file
curDirHead	contiene il Directory Header Block corrente. Per creare questa informazione si utilizza GetDirHead
interleave	questa variabile globale fissa il numero di blocchi di separazione fra un blocco e il successivo quando la routine alloca i settori del disco. Impostando opportunamente il numero di settori da saltare fra la memorizzazione di un blocco e il successivo si può raggiungere una grande efficienza. Per le routine di scrittura del turbo del 1541 il numero più adatto è otto. Con un interleave di otto settori si ottiene la massima velocità di salvataggio dei dati e questa scelta è tanto più conveniente quanto più vasti sono i gruppi di dati con cui si lavora. Non risulta invece molto efficiente per le routine di lettura del turbo. Per i file che devono essere letti frequentemente, l'interleave che rende massima l'efficienza è 9. Questo naturalmente danneggia l'efficienza nelle procedure di scrittura, e si deve quindi fare una scelta diversa caso per caso. GEOS imposta la variabile globale interleave con il valore di default 8

Restituisce:

r2	numero di blocchi allocati
r3	indirizzo T/S dell'ultimo blocco allocato
x	codice d'errore da disco

	0	l'operazione ha avuto successo
	INSUFF_SPACE	questo errore indica che sul disco non sono presenti abbastanza blocchi liberi. Se si verifica un errore da disco, di qualsiasi tipo, la routine interrompe la propria esecuzione, registrando le modifiche fatte fino ad allora nella copia della BAM memorizzata in curDirHead. Dal momento che la routine agisce sulla copia della BAM in memoria e non trasferisce il Directory Header Block nuovamente su disco, le applicazioni possono anche utilizzarla per sapere in anticipo se vi sono abbastanza blocchi liberi su disco

curDirHead	la BAM contenuta nella copia del Directory Header Block, memorizzata nel buffer curDirHead, viene restituita con l'aggiornamento sui nuovi settori allocati. Sino a questo momento il disco non ha subito alcuna variazione, dal momento che la routine non riscrive su disco la nuova BAM. È compito dell'applicazione, dopo l'esecuzione di BlkAlloc, riscrivere su disco l'intero Directory Header Block tramite la routine PutDirHead
fileTrScTab	la routine riporta in questa tavola gli indirizzi T/S dei settori che ha allocato sulla copia della BAM in memoria. Questa tavola è puntata da r6 prima dell'esecuzione della routine e quindi può anche trovarsi a un indirizzo diverso da fileTrScTab
r8L	numero di byte che andranno memorizzati nell'ultimo blocco
r6	puntatore all'inizio del buffer che contiene la tavola degli indirizzi T/S dei blocchi allocati (inalterato)

Distrugge: a, x, y, r4 - r5

Sinossi: BlkAlloc si utilizza per stabilire una concatenazione di blocchi allocabile su disco. Attenzione: la routine non li alloca realmente e lascia inalterata la BAM del disco. Essa altera esclusivamente la copia della BAM mantenuta in memoria e crea il percorso della concatenazione. Tramite questa routine le applicazioni possono sapere in anticipo quanto spazio è disponibile su disco per un determinato file. Il percorso della concatenazione, cioè gli indirizzi T/S dei blocchi allocabili, è memorizzato nel buffer puntato da r6 (generalmente questo buffer è fileTrScTab).

BlkAlloc inizia la ricerca dei blocchi allocabili partendo dalla traccia 1, e si muove verso la traccia 35. Nel corso della ricerca, la traccia 18 viene ignorata. La routine cerca di allocare i blocchi (nella copia della BAM), mantenendo fra i settori l'interleave stabilito dall'applicazione. Una buona scelta dell'interleave accresce la velocità d'accesso al disco. Prima di allocare i settori (nella copia della BAM), la routine si assicura che ci sia abbastanza spazio sul disco.

La prima word puntata da r6, di solito fileTrScTab+2, a operazione ultimata contiene l'indirizzo T/S del primo blocco libero allocato (la routine non altera r6, in modo che l'applicazione possa utilizzarlo ancora). La seconda word individua l'indirizzo T/S del secondo blocco libero allocato, e così via. Le routine che utilizzano questa tavola di indirizzi T/S per trasferire dati su disco, memorizzano nei primi due byte del blocco l'indirizzo T/S del blocco successivo in modo da creare una concatenazione di blocchi conforme allo standard del disk drive 1541. BlkAlloc, dopo aver memorizzato nel buffer l'indirizzo T/S dell'ultimo blocco, scrive nella word successiva il valore 0 e il numero dei byte significativi contenuti nell'ultimo blocco.

Ricordiamo ancora come operano le routine di salvataggio dei dati quando completano la memorizzazione su disco dell'ultimo blocco (all'indirizzo specificato dalla tavola dei blocchi concatenati, come può essere per esempio fileTrScTab). Scrivono, nei primi due byte di questo blocco, il contenuto della word che segue l'ultimo indirizzo T/S nella tavola dei blocchi allocati; nei due byte della word sono contenuti rispettivamente uno 0 per indicare che non seguono altri blocchi, e un numero che indica quanti sono i byte significativi memorizzati nell'ultimo blocco.

BlkAlloc è in grado di allocare su disco un massimo di 127 blocchi alla volta (32.258 byte). Nel caso in cui l'applicazione abbia bisogno di aggiungere dati a un file, o record, già memorizzato su disco, si rende necessario utilizzare la routine NxtBlkAlloc, che aggiunge blocchi a un file.

NxtBlkAlloc

Funzione: Questa routine ha la stessa funzione di BlkAlloc, con la differenza che permette di specificare da quale settore del disco dev'essere iniziata la ricerca dei settori liberi. È una caratteristica che si può sfruttare per aumentare le dimensioni di file già residenti su disco, mantenendo l'interleave ottimale fra i settori.

Indirizzo: $C24D

Chiama: CalcBlksFree, SetNextFree

Parametri:

r3	l'indirizzo T/S del settore dal quale la routine, aggiungendo l'interleave, inizia a cercare il primo settore libero allocabile. Di solito r3 contiene l'indirizzo T/S dell'ultimo blocco allocato in una preesistente concatenazione su disco
r2	numero di byte per i quali la routine deve allocare spazio su disco. Il valore massimo ammesso è 32.258
r6	puntatore all'inizio del buffer che contiene la tavola degli indirizzi T/S dei blocchi allocabili generata dalla routine (r6 di norma punta il buffer fileTrScTab)
curDirHead	contiene il Directory Header Block corrente. È necessario utilizzare la routine GetDirHead per leggerlo da disco
interleave	questa variabile globale fissa il numero di blocchi di separazione fra un blocco e il successivo quando la routine alloca i settori del disco. Impostando opportunamente il numero di settori da saltare si può raggiungere una grande efficienza. Per le routine di scrittura del turbo del 1541 il numero più adatto è otto. Con un interleave di otto settori si ottiene la massima velocità di salvataggio dei dati e questa scelta è tanto più conveniente quanto più vasti sono i gruppi di dati con cui si lavora. Non risulta invece molto efficiente per le routine di lettura del turbo. Per i file che devono essere letti frequentemente, l'interleave che rende massima l'efficienza è 9. Questo naturalmente danneggia l'efficienza nelle procedure di scrittura, e si deve quindi fare una scelta diversa caso per caso. GEOS imposta la variabile globale interleave con il valore di default 8

Restituisce:

r2	numero di blocchi allocati		
r3	indirizzo T/S dell'ultimo blocco libero allocato		
x	codice d'errore da disco		
	0	l'operazione ha avuto successo	
	INSUFF_SPACE	questo errore indica che sul disco non sono presenti abbastanza blocchi liberi. Se si verifica un errore da disco, di qualsiasi tipo, la routine interrompe la propria esecuzione, registrando le modifiche fatte fino ad allora nella copia della BAM memorizzata in curDirHead. Dal momento che la routine agisce sulla copia della BAM in memoria e non trasferisce il Directory Header Block nuovamente su disco, le applicazioni possono anche utilizzarla per sapere in anticipo se vi sono abbastanza blocchi liberi su disco	
curDirHead	la BAM contenuta nella copia del Directory Header Block, memorizzata nel buffer curDirHead, viene restituita con l'aggiornamento sui nuovi settori allocati		
fileTrScTab	la routine riporta in questa tavola gli indirizzi T/S dei settori che ha allocato sulla copia della BAM in memoria. Questa tavola è puntata da r6 prima dell'esecuzione della routine e quindi può anche trovarsi a un indirizzo diverso da fileTrScTab		
r8L	numero di byte che saranno memorizzati nell'ultimo blocco		

Distrugge: a, x, y, r0 - r5

Sinossi: NxtBlkAlloc viene normalmente utilizzata per allocare nuovi blocchi su disco da aggiungere a una concatenazione preesistente. NxtBlkAlloc riceve tra i parametri l'indirizzo T/S dell'ultimo blocco del file, o del record, e da questo punto inizia la ricerca dei settori liberi (creando la tavola degli indirizzi T/S dei blocchi addizionali). Aggiorna inoltre la copia della BAM memorizzata nel buffer curDirHead. Durante la ricerca dei blocchi liberi, la traccia 18 viene ignorata. NxtBlkAlloc, come BlkAlloc, non produce alcuna variazione su disco. È compito dell'applicazione, dopo l'esecuzione della routine, riscrivere la nuova BAM su disco tramite la routine PutDirHead.

Come avviene per BlkAlloc, NxtBlkAlloc è in grado di allocare un massimo di 127 blocchi alla volta (32.258 byte). Per maggiori dettagli si riveda la descrizione di BlkAlloc.

SetNextFree

Funzione: La routine riceve in input l'indirizzo T/S di un blocco e il valore dell'interleave fra settori, trova il blocco e lo contrassegna come allocato nella copia della BAM mantenuta nel buffer dirCurHead.

Indirizzo: $C292

Chiamata da: BlkAlloc, NxtBlkAlloc, SetGEOSDisk

Parametri:

r3 — indirizzo T/S del blocco dal quale la routine inizia la ricerca del successivo settore libero. Se l'indirizzo T/S indica la traccia 18, la routine sa che deve allocare un blocco per la directory e limita la ricerca esclusivamente alla traccia 18. In caso contrario questa traccia è completamente ignorata dalla routine nel corso della ricerca

curDirHead — deve contenere il Directory Header Block del disco, dove si trova la copia della BAM su cui la routine effettua le dovute variazioni

interleave — questa variabile globale fissa il numero di blocchi di separazione fra un blocco e il successivo quando la routine alloca i settori del disco. Impostando opportunamente il numero di settori da saltare si può raggiungere una grande efficienza. Per le routine di scrittura del turbo del 1541 il numero più adatto è otto. Con un interleave di otto settori si ottiene la massima velocità di salvataggio dei dati e questa scelta è tanto più conveniente quanto più sono vasti i gruppi di dati con cui si lavora. Non risulta invece molto efficiente per le routine di lettura del turbo. Per i file che devono essere letti frequentemente, l'interleave che rende massima l'efficienza è 9. Questo naturalmente danneggia l'efficienza nelle procedure di scrittura, e si deve quindi fare una scelta diversa caso per caso. GEOS imposta la variabile globale interleave con il valore di default 8

Restituisce: r3 indirizzo T/S del blocco libero allocato

x codice d'errore da disco

0	l'operazione ha avuto successo
INSUFF_SPACE	questo errore indica che sul disco non sono presenti abbastanza blocchi liberi. Se si verifica un errore da disco, di qualsiasi tipo, la routine interrompe la propria esecuzione, registrando le modifiche fatte fino ad allora nella copia della BAM memorizzata in curDirHead. Dal momento che la routine agisce sulla copia della BAM in memoria e non trasferisce il Directory Header Block nuovamente su disco, le applicazioni possono anche utilizzarla per sapere in anticipo se vi sono abbastanza blocchi liberi su disco

curDirHead la BAM contenuta nella copia del Directory Header Block, memorizzata nel buffer curDirHead, viene restituita con l'aggiornamento sui nuovi settori allocati. Sino a questo momento il disco non ha subito alcuna variazione, dal momento che la routine non riscrive su disco la nuova BAM. È compito dell'applicazione, dopo l'esecuzione di BlkAlloc, riscrivere su disco l'intero Directory Header Block tramite la routine PutDirHead

Distrugge: a, y, r6, r7, r8H

Sinossi: SetNextFree cerca un settore libero da allocare iniziando la ricerca dal settore indicato. Durante la ricerca la routine prova a mantenere l'interleave indicato dal codice di chiamata. Una volta che ha identificato un blocco libero, lo contrassegna come allocato nella copia della BAM mantenuta nel buffer curDirHead, e restituisce il suo indirizzo T/S. SetNextFree viene utilizzata da BlkAlloc, oppure da NxtBlkAlloc, per allocare una serie di blocchi separati da un interleave ideale. Una scelta accurata dell'interleave è necessaria per rendere ottimale la velocità di accesso al file, o record.

FindBAMBit

Funzione: Restituisce lo stato (allocato o disallocato) del settore indicato dall'indirizzo T/S specificato.

Indirizzo: $C2AD

Parametri: r6 indirizzo T/S del settore di cui si vuole conoscere lo stato

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce: A eccezione del flag Z del PSW, queste variabili riguardano solo il disk drive 1541

- a — il byte che contiene il bit corrispondente al settore indicato. Tutti gli altri bit sono azzerati
- Z — riporta lo stato del blocco e questa informazione vale per tutti i tipi di drive:
  - 1 il settore è disallocato
  - 0 il settore è allocato
- r7H — offset all'interno del buffer curDirEntry che indica il byte contenente il numero di blocchi liberi presenti nella traccia indicata
- x — offset all'interno del buffer curDirEntry che indica il byte contenente il bit che corrisponde al settore indicato
- r8H — questo byte è la maschera per isolare il bit corrispondente al settore indicato, all'interno del byte della BAM puntato da x

Distrugge: a, x, r7H, r8H

Sinossi: FindBAMBit, ricevendo l'indirizzo T/S di un settore del disco, ne restituisce lo stato decodificando le informazioni contenute nella BAM. Se il settore è disallocato, il flag Z del PSW viene impostato a 1.

Successive versioni del Kernel saranno probabilmente in grado di gestire più di una BAM alla volta, come dovrebbe accadere per esempio nel caso di drive in grado di leggere dischi a doppia faccia. Le informazioni restituite da questa routine, se si eccettua il flag Z, sono significative solo se il disk drive è il 1541, e devono essere ignorate se vengono usati drive differenti.

FreeBlock

Funzione: Disalloca un singolo settore del disco contrassegnandolo opportunamente nella copia della BAM contenuta nel buffer curDirHead.

Indirizzo:
$C2B9 per GEOS V1.3 e superiori
$9844 per GEOS V1.2

Chiama: FindBAMBit

Parametri:
curDirHead contiene la copia del Directory Header Block del disco
r6 indirizzo T/S del settore da disallocare

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere stata trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:
curDirHead la copia della BAM mantenuta in questo buffer è aggiornata per indicare che il settore è stato disallocato
x codice d'errore da disco

Distrugge: a, r7H, r8H

Sinossi: Avendo a disposizione il drive inizializzato e l'indirizzo T/S del settore, la routine disalloca il blocco e, nella copia della BAM, imposta a 1 il bit di contrassegno che caratterizza lo stato del blocco (allocato, disallocato). Questa routine è riportata nella tavola delle jump-call soltanto a partire dalla versione 1.3 di GEOS. L'applicazione deve controllare quale versione del Kernel di GEOS ha a disposizione, perché nel caso della versione 1.2 questa routine può essere eseguita solo chiamando il suo indirizzo direttamente all'interno del Kernel ($9844) e non ricorrendo alla jump table.

SetGDirEntry

Funzione: SetGDirEntry costruisce il File Entry di un file nel buffer dirEntryBuf e lo memorizza nel primo spazio libero che incontra nella pagina della directory indicata in r10L. Attenzione: questa routine dev'essere utilizzata con dischi in formato GEOS (vedere CkDkGEOS e SetGEOSDisk). SetGDirEntry viene spesso eseguita da SaveFile.

Indirizzo: $C1F0

Chiamata da: SaveFile

Chiama: BldGDirEntry, GetFreeDirBlk

Parametri:

r6	punta al buffer fileTrScTab che contiene la tavola degli indirizzi T/S dei blocchi allocati
r9	punta al buffer fileHeaderBlock che contiene una copia del blocco File Header del file. I primi due byte del blocco File Header, all'atto della memorizzazione su disco, contengono rispettivamente i valori $00 e $FF. Però, finché il blocco resta in memoria questi due byte devono contenere l'indirizzo della stringa a terminazione nulla che riporta il nome del file. Dal momento che i valori per i primi due byte di un blocco File Header non variano mai, è la routine che all'atto della memorizzazione sostituisce il puntatore alla stringa del nome con i valori $00 e $FF. Vedere il capitolo 13 per maggiori dettagli sulla struttura del File Header di un file
r10L	numero della pagina della directory all'interno della quale la routine tenta la memorizzazione del File Entry
curDirHead	contiene la copia del Directory Header Block del disco corrente
fileTrScTab	contiene il percorso dei blocchi appositamente allocati per contenere il file, o il record

Restituisce:

r6	contiene un puntatore che individua lungo il buffer fileTrScTab l'indirizzo T/S del primo blocco allocato per ricevere i dati del file. Si tratta di un parametro creato dalla routine per essere poi passato a WriteFile. Di solito, la prima word del buffer fileTrScTab individua l'indirizzo in memoria del blocco File Header, e la seconda individua l'indirizzo T/S del primo blocco

del file (o del blocco indice nei file a struttura VLIR)

dirEntryBuf contiene il File Entry creato prelevando i dati dal blocco File Header. Questi dati includono l'ora e la data dell'ultimo intervento sul file, e i puntatori al blocco File Header e al primo blocco dati del file (se il file è a struttura VLIR questo primo blocco contiene la tavola indice del file)

diskBlkBuf questo buffer contiene il blocco della directory all'interno del quale la routine ha memorizzato il nuovo File Entry. Il blocco modificato è stato memorizzato su disco

curDirHead dal momento che la routine può trovarsi nella condizione di dover aggiungere un blocco alla directory, la BAM può subire delle modifiche. Questo buffer contiene la copia della BAM modificata

Nota: il contenuto del buffer curDirHead dev'essere riscritto su disco nel caso che la routine abbia dovuto allocare un nuovo blocco per la directory chiamando la routine GetFreeDirBlk. SetGDirEntry non effettua questa operazione

Distrugge: a, y, r0 - r5, r7, r8

Sinossi: SetGDirEntry crea il File Entry per un file e lo memorizza su disco. Normalmente SetGDirEntry è chiamata dalla routine SaveFile nell'ambito del processo di salvataggio di un file su disco. Dovrebbe essere già stata eseguita BlkAlloc, per allocare su disco lo spazio necessario al file e per creare nel buffer fileTrScTab la tavola degli indirizzi T/S dei blocchi allocati.

Dopo aver ricevuto attraverso r10L il numero della pagina della directory nella quale si desidera memorizzare il File Entry del file, SetGDirEntry chiama GetFreeDirBlk per allocare un File Entry in quella pagina (ogni pagina raffigurata in deskTop contiene al massimo otto icone di file, e un blocco della directory contiene al massimo otto File Entry. Non è una coincidenza: pagine e blocchi, in questo contesto, sono interscambiabili). Se la pagina indicata è già piena, GetFreeDirBlk estende la ricerca alle pagine successive, e se necessario alloca un nuovo blocco da aggiungere alla directory. Se tutte le pagine della directory sono piene, GetFreeDirBlk restituisce un errore che viene a sua volta restituito da SetGDirBlk.

GetFreeDirBlk copia nel buffer diskBlkBuf il blocco della directory nel quale ha trovato lo spazio per un File Entry. SetGDirBlk preleva dal blocco File Header (quello del file residente in memoria e puntato da r9) tutte le informazioni che occorrono per la creazione del File Entry, ad eccezione dell'ora e della data correnti. Sulla base di queste informazioni, SetGDirEntry

chiama BldGDirEntry per costruire il File Entry nel buffer dirEntryBuf.

BldGDirEntry alloca il primo dei settori elencati nella tavola degli indirizzi (fileTrScTab) per memorizzarvi il blocco File Header. Se il file è a struttura VLIR, il secondo blocco del concatenamento viene allocato per contenere la tavola indice del file. Il registro r6 inizialmente puntava alla prima locazione nel buffer fileTrScTab, mentre ora deve puntare alla word del buffer che contiene l'indirizzo T/S del primo blocco disponibile su disco per ricevere i dati. Quindi r6 viene incrementato di 2 nel caso di un normale file a struttura SEQUENTIAL, mentre l'incremento sale a 4 nel caso di un file a struttura VLIR. Riassumendo: r6 individua la word che contiene l'indirizzo T/S del primo blocco disponibile. GetGDirEntry memorizza i valori $00 e $FF rispettivamente nei byte 0 e 1 del blocco File Header all'atto della memorizzazione su disco (questi due byte non sono significativi per un blocco File Header).

SetGDirEntry copia nel buffer diskBlkBuf il File Entry appena realizzato, e scrive la data corrente e l'ora nel blocco File Header ancora in memoria. Infine riscrive su disco il blocco della directory contenente il nuovo File Entry.

Nota: SetGDirEntry non scrive su disco i blocchi File Header, Directory Header, e (se VLIR) tavola indice. Si limita ad aggiornare, su disco, il blocco della directory che contiene il File Entry.

Il Directory Header Block conservato in memoria potrebbe aver subito delle variazioni in seguito all'aggiunta di un blocco alla directory, e quindi dovrebbe essere riscritto su disco appena dopo l'esecuzione della routine SetGDirEntry.

BldGDirEntry

Funzione: Di solito questa routine è chiamata da SetGDirEntry, che a sua volta è chiamata da SaveFile. BldGDirEntry crea in memoria il File Entry di un file prelevando le informazioni dal blocco File Header che dovrebbe anch'esso trovarsi in memoria.

Indirizzo: $C1F3

Parametri:

r6	puntatore alla tavola degli indirizzi T/S memorizzata nel buffer fileTrScTab
r9	puntatore al blocco File Header memorizzato nel buffer fileHeader. I primi due byte di questo blocco dovrebbero contenere i valori $00 e $FF. Quando l'applicazione chiama questa routine, i primi due byte devono contenere un puntatore alla stringa del nome del file in memoria. Consultare il capitolo 13 per maggiori dettagli sulla struttura del blocco File Header
fileTrScTab	contiene la lista degli indirizzi T/S dei settori allocati per contenere il file

Restituisce:

r6	contiene un puntatore che individua lungo il buffer fileTrScTab l'indirizzo T/S del primo blocco allocato per ricevere i dati del file. Si tratta di un parametro creato dalla routine per essere poi passato a WriteFile. Di solito, la prima word del buffer fileTrScTab individua l'indirizzo in memoria del blocco File Header, e la seconda individua l'indirizzo T/S del primo blocco del file (o del blocco indice nei file a struttura VLIR)
dirEntryBuf	contiene il File Entry creato prelevando i dati dal blocco File Header. Questi dati includono l'ora e la data dell'ultimo intervento sul file, e i puntatori al blocco File Header e al primo blocco dati del file (se il file è a struttura VLIR questo primo blocco contiene la tavola indice del file)

Distrugge: a, y, r0 - r5, r7, r8

Sinossi: BldGDirEntry costruisce in memoria il File Entry di un file. Il File Entry viene memorizzato nei 30 byte del buffer dirEntryBuf. Le informazioni necessarie alla sua creazione sono prelevate dal blocco File Header, anch'esso

conservato in memoria. Questa routine si utilizza nella procedura di salvataggio di un file su disco. Dovrebbe essere sempre eseguita dopo una chiamata a BlkAlloc, che alloca lo spazio su disco e crea la tavola fileTrScTab, e a GetFreeDirBlk, che alloca un File Entry libero su disco.

In fileTrScTab la prima word fornisce l'indirizzo al quale BldGDirEntry alloca il blocco File Header del file. Nel caso di una struttura VLIR, il blocco indirizzato dalla seconda word in fileTrScTab viene allocato per contenere la tavola indice. Il registro r6 che inizialmente puntava alla prima locazione nel buffer fileTrScTab, ora deve puntare alla word del buffer che contiene l'indirizzo T/S del primo blocco disponibile su disco per ricevere i dati. Quindi r6 viene incrementato di 2 nel caso di un normale file a struttura SEQUENTIAL, mentre l'incremento sale a 4 nel caso di un file a struttura VLIR. Riassumendo: r6 individua la word che contiene l'indirizzo T/S del primo blocco disponibile. GetGDirEntry memorizza i valori $00 e $FF rispettivamente nei byte 0 e 1 del blocco File Header all'atto della memorizzazione su disco (questi due byte non sono significativi per un blocco File Header). BldGDirEntry restituisce nel buffer dirEntryBuf il File Entry completo del file.

FollowChain

Funzione: Dopo aver ricevuto l'indirizzo T/S di un blocco appartenente a una concatenazione memorizzata su disco, restituisce in memoria una tavola che riporta tutti gli indirizzi T/S dei blocchi successivi.

Indirizzo: $C205

Parametri:

r1	indirizzo T/S del blocco della concatenazione dal quale iniziare l'operazione
r3	puntatore al buffer nel quale la routine deve memorizzare la lista degli indirizzi T/S che genera. Dí norma r3 punta a fileTrScTab

Restituisce:

r1	indirizzo T/S dell'ultimo blocco incontrato lungo la concatenazione
r2	inalterato
diskBlkBuf	contiene una copia dell'ultimo blocco del file

Distrugge: a, y, r1, r4

Sinossi: Questa routine riceve un puntatore T/S che indica il blocco di una concatenazione, e crea una tavola in memoria che riporta tutti gli indirizzi T/S dei blocchi che seguono, lungo la concatenazione, quello indicato come parametro. Ogni indirizzo T/S è memorizzato in una word, e la tavola puntata da r3 elenca tutte le word significative. Di solito la tavola è memorizzata nel buffer fileTrScTab.

FastDelFile

Funzione: Cancella un file con un solo accesso quando è già disponibile una lista completa degli indirizzi T/S dei blocchi di cui il file è composto.

Indirizzo: $C244

Chiama: GetDirHead, FreeBlock, PutDirHead

Parametri:

r0	puntatore alla stringa a terminazione nulla che contiene il nome del file da cancellare
r3	puntatore alla lista degli indirizzi T/S dei blocchi da disallocare, in genere fileTrScTab
curDrive	il numero di dispositivo del drive corrente
fileTrScTab	contiene normalmente la lista degli indirizzi T/S dei blocchi da disallocare che costituiscono il file

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce:

x	codice d'errore da disco
curDirHead	la BAM aggiornata contenuta nel Directory Header Block in memoria. La routine riscrive l'intero blocco su disco
turbo	il codice turbo è disattivato, ma non è cancellato dalla memoria interna del drive
dirEntryBuf	il File Entry del file cancellato viene conservato in memoria a causa della chiamata alla routine FindFile

Blocco Directory	Il blocco della directory contente il File Entry del file cancellato viene riscritto su disco con il File Entry azzerato
Blocco File Header	cancellato dal disco insieme al resto del file
Blocco Tavola Indice	cancellato dal disco insieme al resto del file

Distrugge: a, y, r0 - r9

Sinossi: La routine DeleteFile (illustrata nel capitolo precedente) richiede come unico parametro il nome del file da cancellare, ma deve effettuare molti accessi al disco ed è quindi abbastanza lenta. In genere, quando si manipola un file si è in possesso di molte informazioni, oltre al nome del file. Se l'applicazione, fra le varie informazioni che possiede sul file, conserva anche la tavola fileTrScTab, può utilizzare per la cancellazione FastDelFile. Questa routine richiede come parametri il nome del file (per cercare e cancellare il File Entry da disco), e la lista dei blocchi da disallocare. Oltre a queste informazioni, occorrono ancora un accesso alla BAM per disallocare i blocchi del file e un accesso alla directory per cancellare il File Entry. Le applicazioni, se sono in possesso di questi dati, possono cancellare i file con un notevole risparmio di tempo, dal momento che FastDelFile è rapidissima. La lista dei blocchi memorizzata nel buffer puntato da r3 può essere generata dalla routine FollowChain.

FreeFile

Funzione: Dopo aver ricevuto il File Entry di un file, la routine disalloca tutti i settori occupati dal file su disco, ma lascia inalterato il File Entry nella directory.

Indirizzo: $C226

Chiamata da: DeleteFile

Chiama: GetDirHead, PutDirHead

Parametri:

dirEntryBuf	File Entry del file
r9	puntatore al buffer dirEntryBuf

Preparazione del drive: Per accedere al disco con successo, il drive dev'essere inizializzato. Il byte d'identificazione ID del disco dev'essere memorizzato nella RAM di sistema del drive. La variabile curDrive deve contenere il numero di dispositivo del drive corrente, e viene aggiornata tramite una chiamata alla routine SetDevice. Anche NewDisk o OpenDisk devono essere già state chiamate per impostare altre variabili. In alcuni casi il programmatore può decidere di utilizzare routine di livello inferiore, se osserva che le variabili necessarie sono già state impostate o aggiornate da routine di alto livello. Dal momento che scendendo di livello le routine diventano molto più rapide nell'esecuzione, in questo modo il programmatore può ottimizzare notevolmente gli accessi al disco.

Restituisce:

x	codice d'errore da disco
curDirHead	contiene la BAM aggiornata. È la copia del Directory Header Block riscritto su disco
disco	la BAM viene aggiornata
disco	il blocco della directory che contiene il File Entry del file cancellato non viene alterato
disco	il blocco File Header del file viene cancellato con il resto del file
disco	il blocco indice, se il file è a struttura VLIR, viene cancellato insieme al resto del file

Distrugge: a, y, r0 - r9

Sinossi: FreeFile riceve il File Entry di un file e quindi agisce sulla BAM del disco disallocando tutti i settori del file, ma lasciando inalterato il File Entry nella directory. La routine funziona per ogni tipo di file, compresi quelli a struttura VLIR. In quest'ultimo caso la routine cancella anche la tavola indice e tutti i record del file. Se si desidera una cancellazione completa, che comprenda anche il File Entry, si deve utilizzare la routine DeleteFile. FreeFile è una subroutine di DeleteFile.

ChangeDiskDevice

Funzione: Ordina al disk drive di cambiare il suo numero d'accesso sul bus seriale.

Indirizzo: $C2BC

Parametri: a — nuovo numero di dispositivo: 8, 9, 10 o 11

Restituisce: x — stato dell'errore da disco, x = 0 corrisponde a nessun errore. Consultare l'appendice per una lista completa degli errori da disco

Distrugge: a, x, r1

Sinossi: ChangeDiskDevice viene chiamata da deskTop, per esempio quando si richiede di aggiungere un drive. Questa routine comunica al drive un nuovo numero di accesso al bus seriale.

StartAppl

Funzione: Inizializza il sistema e cede il controllo all'applicazione correntemente in memoria.

Indirizzo: $C22F

Parametri: r7 indirizzo della routine d'inizializzazione dell'applicazione alla quale si desidera cedere il controllo

Restituisce: Niente

Distrugge: In genere tutti i registri

Sinossi: Questa routine procede a inizializzare il sistema tramite l'operazione di "partenza a caldo" (Warm Start) descritta nel capitolo 20 e cede il controllo all'indirizzo contenuto in r7. Questo indirizzo deve individuare la routine d'inizializzazione dell'applicazione.

16 ROUTINE DI LIVELLO PRIMITIVO

Nel capitolo 15 abbiamo esaminato le routine di livello intermedio che GEOS mette a disposizione delle applicazioni. Consideriamo ora alcune routine di livello primitivo che formano la base di lavoro su cui è stato sviluppato il sistema dei file di GEOS. Eccone l'elenco:

InitForIO
DoneWithIO
PurgeTurbo
EnterTurbo
ExitTurbo
ReadBlock
WriteBlock
VerWriteBlock

InitForIO

Funzione: Questa routine dev'essere chiamata prima di ogni accesso al bus seriale. Ha il compito di disabilitare gli interrupt e gli sprite, di attivare la ROM del Kernel del C-64 e lo spazio di I/O, e infine di aggiornare alcune variabili per le comunicazioni seriali.

Indirizzo: $C25C

Parametri: Nessuno

Restituisce: Gli interrupt e gli sprite disabilitati. La routine salva temporaneamente il loro stato, forza l'esecuzione di un NMI (ma fa in modo che dopo resti inattivo), disattiva l'IRQ, attiva la ROM del Kernel del C-64 e lo spazio di I/O

Distrugge: a

Sinossi: Si deve chiamare questa routine prima di compiere qualsiasi operazione di accesso al bus seriale. Ogni funzione ordinariamente gestita dal sistema, come gli interrupt e gli sprite, viene sospesa. Viene salvata la configurazione corrente dei banchi di memoria, e vengono attivati il Kernel del C-64 e lo spazio di I/O. InitForIO aggiorna inoltre i vettori IRQ e NMI, puntandoli a routine fittizie che non svolgono nessuna operazione; configura la porta seriale nel suo stato consueto; imposta un contatore che si decrementa generando un NMI in modo che la linea NMI sia mantenuta a livello basso sino a quando non viene eseguita la routine DoneWithIO; disattiva le chiamate di interrupt generate dal raster; salva il registro che contiene i flag di stato degli sprite e infine disabilita gli sprite.

DoneWithIO

Funzione: Questa routine dev'essere chiamata quando si conclude l'accesso al bus seriale. Riabilita gli sprite, gli interrupt, ripristina la configurazione di memoria precedente alla chiamata di InitForIO e ripristina il normale funzionamento della macchina in ambiente GEOS.

Indirizzo: $C25F

Parametri: Nessuno

Restituisce: Il sistema nella configurazione precedente alla chiamata di InitForIO

Distrugge: a

Sinossi: DoneWithIO ripristina la configurazione di sistema precedente alla chiamata di InitForIO, riabilita gli sprite e gli interrupt di NMI, di IRQ e di raster e riattiva la configurazione dei banchi di memoria precedentemente salvata.

PurgeTurbo

Funzione: Restituisce il controllo del processore del disk drive 1541 alle routine del DOS residenti nella ROM interna al drive. Imposta un flag per indicare al sistema che il codice del turbo non è più residente nel disk drive.

Indirizzo: $C235

Chiamata da: GetBlock, PutBlock, NewDisk, ReadFile

Chiama: InitForIO, DoneWithIO

Parametri:

curDrive — numero del drive per cui dev'essere disattivato il turbo

turboFlags — questa variabile globale contiene i flag che indicano lo stato del codice turbo nel drive 8. Per il drive 9 è turboFlags + 1. La routine vi accede per aggiornare i flag

Bit 7
- 1 — indica che il codice turbo è ancora residente nella RAM del drive
- 0 — indica che il codice turbo non è residente nella RAM del drive

Bit 6
- 1 — indica che il codice turbo nella RAM del drive è in esecuzione
- 0 — indica che il codice turbo nella RAM del drive non è in esecuzione

Restituisce:

x — codice d'errore da disco

turboFlags — questa variabile globale contiene i flag che indicano lo stato del codice turbo nel drive 8. Per il drive 9 è turboFlags + 1.

Bit 7
- 0 — per indicare che il codice turbo non è più residente nel drive

Bit 6
- 0 — per indicare che il codice turbo residente nel drive non è in esecuzione

Distrugge: a, x, y, r0 - r3

Sinossi: Si chiama PurgeTurbo per far cessare l'esecuzione del codice turbo nel drive e indicare (tramite i flag della variabile turboFlags associata al drive selezionato) che questo non è più in esecuzione e non è più residente. In genere le applicazione utilizzano PurgeTurbo quando devono inviare al drive un comando DOS 1541 che potrebbe alterare il contenuto della RAM del drive. Quando il sistema deve comunicare con il drive, accede a turboFlags per sapere qual è lo stato del codice turbo e si comporta di conseguenza.

Per riallocare e riattivare il codice turbo nel drive occorre chiamare la routine EnterTurbo.

EnterTurbo

Funzione: Se il codice turbo non è allocato nella memoria del drive selezionato, la routine lo riscrive e quindi lo attiva. Altrimenti si limita ad attivarlo.

Indirizzo: $C214

Chiamata da: GetBlocks, NewDisk

Chiama: SetDevice, InitForIO, DoneWithIO

Parametri:

curDrive	il numero del drive interessato alla riattivazione del codice turbo
turboFlags	questa variabile globale contiene i flag che indicano lo stato del codice turbo nel drive 8. Per il drive 9 è turboFlags + 1

Bit 7
- 1 indica che il codice turbo è ancora residente nella RAM del drive
- 0 indica che il codice turbo non è residente nella RAM del drive

Bit 6
- 1 indica che il codice turbo nella RAM del drive è in esecuzione
- 0 indica che il codice turbo nella RAM del drive non è in esecuzione

Restituisce:

x	codice d'errore da disco
drive	il codice turbo è in esecuzione

Distrugge: a, y

Sinossi: EnterTurbo ordina al sistema operativo del drive 1541 d'iniziare l'esecuzione del codice turbo. Se i flag della variabile turboFlags indicano che il codice non è residente, la routine lo trasferisce nuovamente nella memoria del drive e quindi lo manda in esecuzione. EnterTurbo viene eseguita ogni volta che si desidera compiere un'operazione con il disco. Il codice turbo, conclusa l'esecuzione della routine, mantiene il possesso del bus seriale. Se si manifesta la necessità di utilizzare il bus seriale per comunicare con un altro

dispositivo, si deve ricorrere alla routine ExitTurbo. La gestione di queste due routine è affidata di solito alle routine di accesso al disco d'alto livello. Tuttavia, le applicazioni che desiderano accedere al drive al livello più basso possono utilizzare queste due routine direttamente. In mancanza di precise esigenze, consigliamo di non lasciare il codice turbo attivo nel drive. È preferibile che l'applicazione lo disattivi, chiamando la routine ExitTurbo, al termine di ogni accesso.

ExitTurbo

Funzione: Disattiva il codice turbo nel drive corrente. La routine non lo disalloca: il codice turbo è ancora presente nel drive e può essere riattivato tramite la routine EnterTurbo.

Indirizzo: $C232

Parametri: curDrive il numero del drive interessato alla disattivazione del codice turbo

Restituisce: Il bit 6 del turboFlag associato al drive impostato a 0

Distrugge: a, x, y, r0 - r3

Sinossi: ExitTurbo disattiva il codice turbo residente nel drive indicato da curDrive, ma non lo disalloca. Il turbo può essere riattivato semplicemente chiamando EnterTurbo.

ReadBlock

Funzione: Svolge la stessa funzione di GetBlock, ma presuppone che il codice turbo nel drive sia già in esecuzione e che gli sprite e gli interrupt siano già disabilitati.

Indirizzo: $C21A

Chiamata da: GetBlock

Parametri:

r1	indirizzo T/S del settore sul disco
r4	indirizzo in memoria del buffer nel quale memorizzare il blocco
curDrive	il numero del drive corrente

Sistema: Il drive dev'essere stato inizializzato tramite OpenDisk o NewDisk, e dev'essere stato selezionato sul bus seriale tramite la routine SetDevice. Lo spazio di I/O del C-64 dev'essere attivo, mentre gli sprite e gli interrupt devono essere stati precedentemente disabilitati.

Restituisce:

x	codice d'errore da disco
r4	puntatore al blocco trasferito in memoria (inalterato)
r1	inalterato

Distrugge: a, y

Sinossi: ReadBlock è la più primitiva routine di accesso al disco di cui GEOS dispone. Quando si leggono o si scrivono lunghe concatenazioni di blocchi su disco, è opportuno minimizzare il tempo di lavoro della testina del drive. Le routine d'alto livello messe a disposizione da GEOS per accedere a lunghe concatenazioni di blocchi, sono relativamente rapide. Ma per il salvataggio dei file si può raggiungere una velocità anche maggiore creando una routine di scrittura che salvi tutto il file ed effettui la verifica in un secondo tempo. Le applicazioni dotate di routine per la gestione dei file che non seguono il normale sistema della concatenazione dei blocchi, possono adottare tecniche diverse per aumentare la velocità di accesso.

ReadBlock dev'essere utilizzata in questo modo:

```
jsr EnterTurbo      ;manda in esecuzione il codice turbo nel drive
jsr InitForIO       ;prepara il sistema per l'accesso al drive
jsr ReadBlock       ;legge il blocco da disco
...                 ;legge altri blocchi
jsr ReadBlock
jsr DoneWithIO      ;riporta il sistema al suo stato normale
```

L'applicazione deve passare alla routine l'indirizzo T/S del blocco da trasferire in memoria. ReadBlock trasferisce il blocco dal disco nel drive al buffer indicato in r4, e legge anche la prima word che contiene l'indirizzo T/S del blocco successivo.

Prendiamo in considerazione un caso reale: supponiamo che l'applicazione memorizzi alcune informazioni chiave nei primi byte contenuti nel primo blocco di ogni record di cui è composto un file a struttura VLIR. Quando possiede in memoria la tavola indice dei record, l'applicazione può agevolmente accedere tramite ReadBlock al primo blocco di ogni record e prelevare le informazioni chiave.

WriteBlock

Funzione: Svolge la stessa funzione di PutBlock, ma presuppone che il codice turbo nel drive sia già in esecuzione e che gli sprite e gli interrupt siano già stati disabilitati.

Indirizzo: $C220

Chiamata da: PutBlock

Parametri:

r1	indirizzo T/S del settore nel quale memorizzare il blocco
r4	puntatore al buffer da 256 byte che contiene il blocco da trasferire su disco, in genere diskBlkBuf

Sistema: Il drive dev'essere stato inizializzato tramite OpenDisk o NewDisk, e dev'essere stato selezionato sul bus seriale tramite la routine SetDevice. Lo spazio di I/O del C-64 dev'essere attivo, mentre gli sprite e gli interrupt devono essere stati precedentemente disabilitati.

Restituisce:

x	codice d'errore da disco
r4	puntatore al buffer (inalterato)
r1	inalterato

Distrugge: a, y

Sinossi: WriteBlock è la routine di scrittura su disco più primitiva di cui GEOS dispone. Richiede che gli interrupt e gli sprite siano disabilitati, e che il codice turbo sia in esecuzione. Perché un file sia memorizzato correttamente, i blocchi devono essere disposti a catena (cioè ogni blocco deve contenere nei primi due byte l'indirizzo T/S del blocco successivo). Quindi la disposizione a catena deve esistere già quando i blocchi si trovano nella memoria del C-64. Di solito il blocco da trasferire viene memorizzato nel buffer diskBlkBuf. WriteBlock dev'essere utilizzata in questo modo:

```
jsr EnterTurbo      ;manda in esecuzione il codice turbo nel drive
jsr InitForIO       ;prepara il sistema per l'accesso al drive
jsr WriteBlock      ;scrive il blocco su disco
...                 ;scrive altri blocchi
jsr WriteBlock
jsr DoneWithIO      ;riporta il sistema al suo stato normale
```

L'applicazione deve passare alla routine l'indirizzo T/S del settore su disco dove trasferire il blocco residente in memoria (che deve contenere nei primi due byte l'indirizzo T/S del blocco successivo). Di solito l'applicazione può servirsi altrettanto bene di una routine d'alto livello, e non ha bisogno di ricorrere direttamente a WriteBlock.

Nel caso si desideri effettuare la verifica del blocco appena trasferito, è sufficiente, subito dopo la chiamata alla routine WriteBlock, eseguire VerWriteBlock. Dal momento che le due routine utilizzano gli stessi parametri e WriteBlock non li altera, possono essere chiamate consecutivamente.

VerWriteBlock

Funzione: Questa routine viene eseguita subito dopo la chiamata a WriteBlock per verificare i dati trasmessi su disco.

Indirizzo: $C223

Parametri:
r1 indirizzo T/S del settore da verificare con i dati in memoria
r4 puntatore al buffer da 256 byte che contiene il blocco da confrontare con quello contenuto nel settore del disco (di solito diskBlkBuf)

Sistema: Il drive dev'essere stato inizializzato tramite OpenDisk o NewDisk, e dev'essere stato selezionato sul bus seriale tramite la routine SetDevice. Lo spazio di I/O del C-64 dev'essere attivo, mentre gli sprite e gli interrupt devono essere stati precedentemente disabilitati.

Restituisce:
x codice d'errore da disco; se la verifica non ha avuto successo, la routine restituisce l'errore WR_VER_ERR (codice $25)
r4 puntatore al buffer (inalterato)
r1 inalterato

Distrugge: a, y

Sinossi: VerWriteBlock è una routine primitiva che viene di solito eseguita in combinazione con WriteBlock per verificare che il blocco sia stato memorizzato su disco correttamente. Dal momento che WriteBlock non altera i parametri che ha ricevuto, subito dopo la sua esecuzione può essere fatta la chiamata jsr VerWriteBlock senza alcuna operazione intermedia. Vediamo un tipico impiego della routine.

```
jsr EnterTurbo      ;manda in esecuzione il codice turbo nel drive
jsr InitForIO       ;prepara il sistema per l'accesso al drive
jsr WriteBlock      ;scrive il blocco su disco
jsr VerWriteBlock   ;verifica la correttezza del trasferimento
jsr DoneWithIO      ;riporta il sistema al suo stato normale
...
```

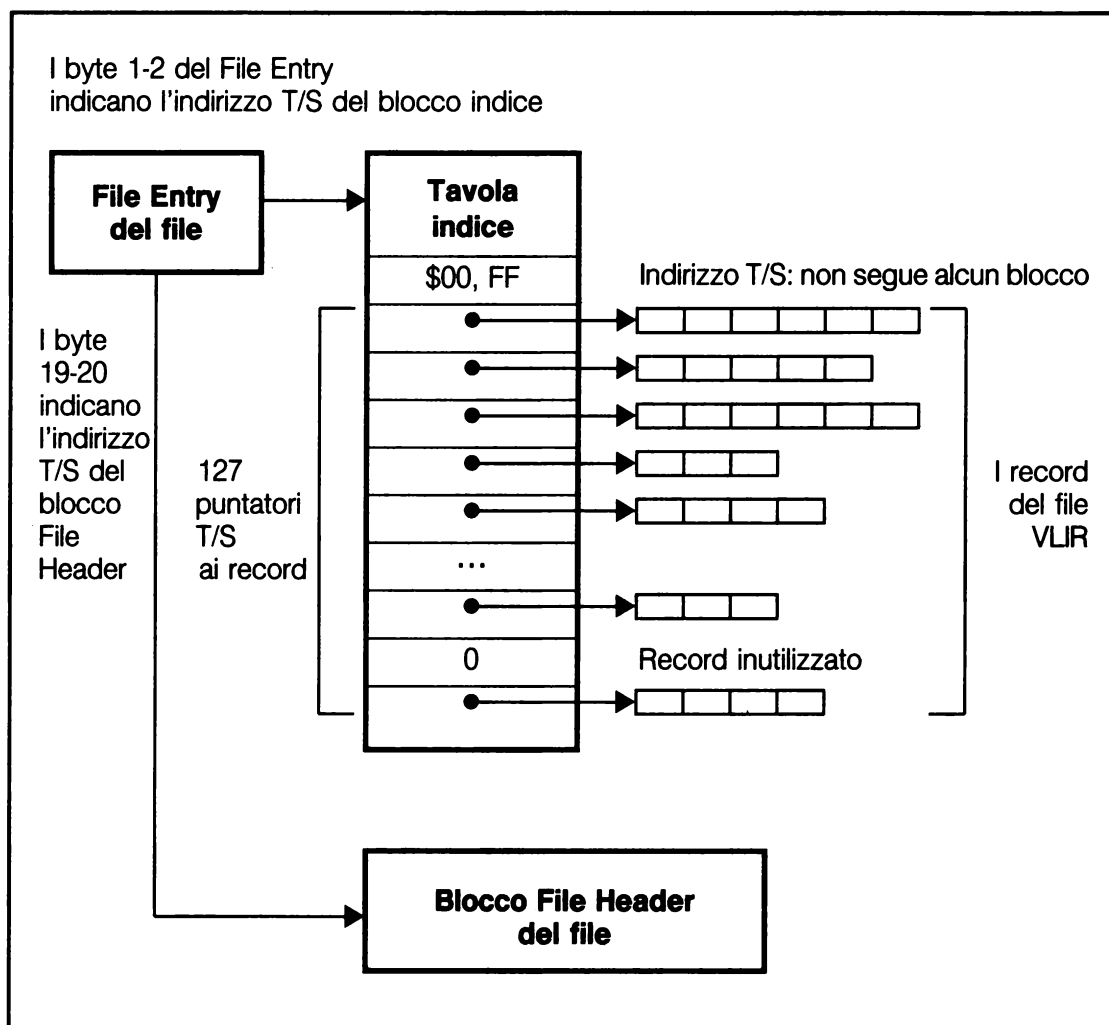
17 I FILE CON STRUTTURA VLIR

La struttura dei file VLIR è stata ideata e realizzata per permettere alle applicazioni di crescere e superare i 30K di memoria disponibili in ambiente GEOS. Grazie all'elevata velocità di accesso al disco che si ottiene con GEOS, diventa particolarmente utile suddividere un'applicazione in più moduli e all'occorrenza richiamarli da disco. Il miglior modo per gestire un'applicazione suddivisa in moduli è mantenere in memoria il modulo principale e richiamare gli altri da disco, allocando per loro lo stesso spazio di memoria, che sarà di volta in volta a disposizione del modulo richiamato. Il modulo residente può chiamare le routine di qualunque altro modulo, ma gli altri possono chiamare solo le routine del modulo residente perché non possono risiedere contemporaneamente in memoria. L'applicazione può richiamare in memoria, in tempi brevissimi, il modulo da utilizzare, eseguire le necessarie operazioni ed eventualmente caricare il modulo successivo. Quando un'applicazione è suddivisa in moduli, si dice che è "gestita in overlay".

Un file a struttura VLIR è un insieme di record, ognuno dei quali può contenere un modulo dell'applicazione. Ogni record non è altro che una normale concatenazione di blocchi sequenziali. Un file a struttura VLIR può essere considerato come una collezione di diversi file. Le stesse routine che vengono utilizzate per salvare su disco un normale file a struttura SEQUENTIAL possono essere impiegate per salvare i record di un file a struttura VLIR. Oltre a queste routine comuni alle due strutture, GEOS contiene diverse altre routine che sono riservate esclusivamente ai file a struttura VLIR.

Le routine di gestione dei file a struttura VLIR allocano i settori del disco per i record facendo riferimento alla tavola degli indirizzi T/S dei settori (memorizzata generalmente nel buffer fileTrScTab), come avviene per i file normali. Ogni record non può quindi essere composto da più di 127 blocchi (32.258 byte), mentre la dimensione minima è di zero blocchi (record vuoto). La dimensione massima è sufficiente per

contenere perfino i codici che arrivano a impiegare la memoria del buffer di schermo, cioè che vengono allocati nello spazio di memoria che va da $0400 a $8000. La tavola indice, che abbiamo già menzionato nei precedenti capitoli, contiene i puntatori T/S che individuano il primo blocco di ogni record. La tavola che segue illustra la struttura fondamentale di un file VLIR, evidenziando l'organizzazione dei record nel file.

Struttura dei file VLIR

VLIR è l'acronimo di Variable Length Indexed Record (Record Indicizzati di Lunghezza Variabile). Sia le applicazioni sia le strutture di dati possono essere salvate in file VLIR. Per esempio, i file delle fonti caratteri (il tipo è FONT) sono file VLIR, nei quali ciascun record contiene i dati di un diverso corpo carattere.

Un file a struttura VLIR può essere facilmente identificato andando a leggere il byte del File Entry che specifica i due possibili tipi: SEQUENTIAL o VLIR. Nel caso di una struttura VLIR, il File Entry contiene un puntatore T/S al blocco della tavola indice. Nel caso di una struttura SEQUENTIAL, questo puntatore indica il primo blocco del file. Per maggiori dettagli sui File Entry, si consulti il capitolo 13.

La tavola indice, memorizzata in un blocco del disco, è in grado di contenere gli indirizzi T/S di 127 blocchi, numerati da 0 a 126. Dal momento che ogni blocco è l'inizio di un record, la tavola indice è in grado di indirizzare 127 record, numerati da 0 a 126. La prima word della tavola indice non viene impiegata, la seconda e le successive contengono i puntatori T/S ai 127 record. Il resto delle informazioni documentate nel File Entry e nel blocco File Header non sono diverse da quelle relative ai file a struttura SEQUENTIAL.

Le routine per la gestione VLIR dei file

GEOS prevede un particolare set di routine per la manipolazione dei file a struttura VLIR. Introduciamo il concetto di "puntatore al record corrente". Dal momento che i record sono numerati, possiamo indicarli alle routine di gestione per mezzo di un puntatore numerico che ne riporta il numero d'ordine. Il record corrente può essere cancellato, letto o riscritto. A ogni accesso, l'applicazione si trova a gestire un intero record. Quindi le applicazioni devono riservare in memoria spazio sufficiente per la manipolazione di un record di dimensione massima (127 blocchi). Ovviamente se l'applicazione è stata realizzata per gestire insiemi di dati – o moduli – di dimensioni inferiori a quella massima, lo spazio riservato in memoria può essere anche molto minore. È possibile anche aggiungere record vuoti, che seguono o precedono un record preesistente nella lista. In effetti, dal momento che ciascuno è identificato da un numero d'ordine, non è necessario che record consecutivi da un punto di vista logico siano contigui nella tavola indice. Il numero d'ordine associato a un record (0 - 126) non è altro che il numero della word della tavola indice che lo indirizza. Per esempio, il record 0 è indirizzato dalla word 0 del blocco tavola indice (la prima word viene considerata in posizione –1). Per il momento non c'è la possibilità di separare un record e riconnetterlo altrove. DeleteRecord è distruttiva dal momento che disalloca tutti i settori del record e cancella il suo indirizzo dalla tavola indice. InsertRecord è in grado di lavorare solo con record vuoti.

Mantenendo in memoria una copia della tavola indice, di solito nel buffer fileHeader, è possibile individuare un record direttamente, tramite la routine PointRecord, anziché scorrere avanti e indietro la lista dei record tramite le routine NextRecord e PreviousRecord.

Nella realizzazione delle routine di gestione dei record è stata posta particolare attenzione alla documentazione degli errori. Il prossimo paragrafo riporta una lista degli errori che si possono verificare manipolando i record di un file a struttura VLIR. I codici d'errore sono sempre restituiti nel registro x. Se una routine restituisce x = 0, significa che non si è verificato alcun errore. Una lista dettagliata dei possibili errori da disco è riportata in appendice.

I messaggi d'errore

UNOPENED_VLIR
Questo errore si verifica quando si cerca di leggere/scrivere/cancellare/appendere un record di un file VLIR prima che il file sia stato aperto tramite la routine OpenRecordFile.

INV_RECORD
Questo errore si verifica quando si cerca di utilizzare una delle routine di lettura, scrittura e spostamento del puntatore con un record che non esiste (nella corrispondente word della tavola indice non è memorizzato alcun indirizzo T/S). Questo errore non è fatale e si può lasciare che continui a verificarsi quando si desidera muovere il puntatore lungo la lista dei record.

OUT_OF_RECORDS
Questo errore si verifica quando si cerca di inserire/appendere un record a un file che già contiene il massimo numero di record concesso (127 record, attualmente).

STRUCT_MISMAT
Questo errore si verifica quando viene utilizzata una routine in grado di manipolare un certo tipo di file per gestire file di tipo diverso.

Creazione di un file a struttura VLIR

Per prima cosa si procede alla creazione di un file VLIR vuoto, ricorrendo alla routine SaveFile. Il File Header da passare a SaveFile dovrebbe contenere le seguenti informazioni:

Tipo del File C-64	USER
Struttura GEOS del File	VLIR
Per i File Dati:	
Indirizzo di Caricamento:	0
Indirizzo di Fine File:	\$FFFF (−1), questo indirizzo insolito segnala a SaveFile che deve creare la struttura di un file VLIR vuoto
Per le Applicazioni:	
Indirizzo di Caricamento:	locazione alla quale dev'essere caricato il primo record del file
Indirizzo di Fine File:	\$FFFF (−1), questo indirizzo insolito segnala a SaveFile che deve creare la struttura di un file VLIR vuoto

Questo è sufficiente per creare su disco un file VLIR con la tavola indice priva di record (vuota). Il puntatore ai record è impostato al valore −1: puntatore nullo. Prima di poter manipolare in qualsiasi modo un record del file, il file dev'essere aperto tramite la routine OpenRecordFile. Questa routine prepara i buffer interni di cui GEOS ha bisogno per manipolare il file. Se il file è completamente vuoto, cioè privo di record, il primo dev'essere creato tramite la routine AppendRecord. Dopo queste operazioni, si può passare all'esecuzione delle altre routine InsertRecord, AppendRecord e DeleteRecord.

Quando l'applicazione non deve più compiere alcuna operazione con i record del file, deve assolutamente chiuderlo tramite la routine CloseRecordFile. Questa routine aggiorna su disco la tavola indice, la BAM e il contatore dei blocchi contenuto nel File Entry. Attualmente GEOS non è in grado di gestire due file VLIR contemporaneamente.

Illustriamo ora nei dettagli le routine che permettono di gestire i file a struttura VLIR.

OpenRecordFile

Funzione: Dopo aver ricevuto il nome di un file VLIR già esistente, questa routine lo apre, rendendolo così accessibile all'applicazione.

Indirizzo: $C274

Chiama: FindFile, GetBlock

Parametri:

r0	puntatore alla stringa a terminazione nulla che contiene il nome del file

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

x	codice d'errore da disco
fileHeader	questo buffer contiene la tavola indice del file
usedRecords	numero di record di cui il file è composto
fileWritten	questo flag indica se il file è stato ulteriormente modificato dopo l'ultima modifica nella BAM e nella tavola indice. Se vale 0, nessuna modifica è stata effettuata
curRecord	puntatore ai record: se il file contiene almeno un record non vuoto, curRecord = 0, altrimenti (file vuoto) curRecord = −1
dirEntryBuf	questo buffer contiene una copia del File Entry del file
curDirHead	questo buffer contiene una copia del Directory Header Block del file

Distrugge: a, y, r1, r4 - r6

Sinossi: OpenRecordFile prepara le variabili globali necessarie alle altre routine di manipolazione dei file VLIR per operare correttamente. OpenRecordFile chiama FindFile per scorrere la directory alla ricerca del file e, se il file viene trovato, verifica che abbia una struttura VLIR e imposta le variabili necessarie.

CloseRecordFile

Funzione: Aggiorna la tavola indice del file VLIR e la BAM del disco. Segnala che non vi è alcun file aperto.

Indirizzo: $C277

Chiama: UpdateRecordFile

Parametri: La routine OpenRecordFile ha preparato le seguenti variabili: usedRecord, curRecord, fileWritten, fileHeader, curDirHead, dirEntryBuf (il buffer fileHeader contiene la tavola indice)

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

tavola indice	aggiornata su disco
BAM	aggiornata su disco
fileWritten	impostato a 0
File Entry	se il file è stato modificato, la routine aggiorna il numero di blocchi usati per il file e aggiorna l'ora e la data copiando i valori correnti dalle variabili di sistema

Distrugge: a, y, r1, r4, r5

Sinossi: La routine chiama UpdateRecordFile per aggiornare la variabili su disco appena menzionate. Se il file è stato in qualche modo modificato, l'ora e la data vengono aggiornate con i valori di sistema correnti. La routine imposta un flag per indicare che non c'è nessun file VLIR aperto.

UpdateRecordFile

Funzione: Aggiorna su disco la tavola indice, la BAM, la data e l'ora associate al file. Il file non viene chiuso.

Indirizzo: $C295

Chiamata da: CloseRecordFile

Chiama: GetBlock, PutBlock

Parametri: La routine OpenRecordFile ha preparato le seguenti variabili: usedRecord, curRecord, fileWritten, fileHeader, curDirHead, dirEntryBuf (fileHeader contiene la tavola indice)

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

tavola indice	aggiornata su disco
BAM	aggiornata su disco
fileWritten	impostato a 0
File Entry	se il file è stato modificato, la routine aggiorna il numero di blocchi usati per il file e aggiorna l'ora e la data copiando i valori correnti dalle variabili di sistema

Distrugge: a, y, r1, r4, r5

Sinossi: UpdateRecordFile aggiorna su disco le variabili appena menzionate. Se il file ha subito qualche modifica, l'ora e la data memorizzate nel File Entry vengono aggiornate con i valori di sistema correnti.

PreviousRecord NextRecord PointRecord

Funzione: Impostano il puntatore ai record in modo che punti rispettivamente al record che precede quello corrente, a quello che lo segue o a un particolare record del file VLIR.

Indirizzi:

PreviousRecord	$C27D
NextRecord	$C27A
PointRecord	$C280

Parametri:

a	contiene il numero del record da puntare nel caso della routine PointRecord. Questo parametro non ha significato per le routine NextRecord e PreviousRecord

La routine OpenRecordFile ha preparato le seguenti variabili: usedRecord, curRecord, fileWritten, fileHeader, curDirHead, dirEntryBuf (fileHeader contiene la tavola indice)

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

curRecord	numero del record corrente
x	flag d'errore nell'operazione: 0 operazione effettuata con successo 1 l'aggiornamento richiesto del puntatore non è stato effettuato perché il puntatore si trova già al limite della tavola indice
y	flag di file pieno: se vale 0 non è un segnale d'errore, ma indica che il record puntato è vuoto. Il valore attuale memorizzato in y rappresenta la traccia del record memorizzata nella tavola indice. Se il record è vuoto (non è composto da alcun blocco) in valore di y è zero
r1	l'indirizzo T/S del primo blocco del record correntemente puntato
fileHeader	inalterato

Distrugge: Niente

Sinossi: PreviousRecord, PointRecord e NextRecord variano il contenuto del puntatore ai record. Se si utilizza PointRecord si deve assegnare come parametro il numero del record.

Muovendo il puntatore può verificarsi un errore, come per esempio avviene quando si chiama NextRecord e il puntatore già punta l'ultimo record. In questo caso viene passato in x il codice dell'errore, e il valore corrente del puntatore non subisce alcuna modifica. Se invece non si verifica alcun errore, viene caricato in r1 l'indirizzo T/S del primo blocco del record corrente, prelevato dalla tavola indice. r1L, la traccia nella quale si trova il primo blocco del record, viene copiata in y. Se y viene restituito con il valore 0, significa che il record correntemente puntato è completamente vuoto.

DeleteRecord

Funzione: Cancella il record correntemente puntato e sposta il puntatore (curRecord) al record successivo.

Indirizzo: $C283

Chiama: GetDirHead

Parametri: La routine OpenRecordFile ha preparato le seguenti variabili: usedRecord, curRecord, fileWritten, fileHeader, curDirHead, dirEntryBuf (fileHeader contiene la tavola indice)

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

x	codice d'errore da disco
curRecord	punta al record successivo, o all'ultimo record nella lista

Distrugge: a, y, r0 - r9

Sinossi: Il record corrente viene cancellato e il puntatore curRecord viene spostato al successivo. Se il record cancellato era l'ultimo del file VLIR, dopo la cancellazione il puntatore punta al nuovo "ultimo record".

WriteRecord

Funzione: Salva una particolare area di memoria all'interno del record corrente.

Indirizzo: $C28F

Chiama: GetDirHead, WriteFile, BlkAlloc

Parametri:

r2	numero di byte da salvare
r7	indirizzo d'inizio in memoria dell'area da salvare

La routine OpenRecordFile ha preparato le seguenti variabili: usedRecord, curRecord, fileWritten, fileHeader, curDirHead, dirEntryBuf (fileHeader contiene la tavola indice)

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

x	codice d'errore da disco
fileHeader	contiene la tavola indice aggiornata
fileSize	dimensione in blocchi del record
fileWritten	caricato con il valore $FF (−1) per indicare che il file, dal momento dell'apertura, ha subito una modifica
fileTrScTab	tavola dei settori utilizzati per salvare il record
disco	il nuovo record salvato su disco
r8L	numero di byte memorizzati nell'ultimo settore
r3	indirizzo T/S dell'ultimo settore allocato su disco
curDirHead	copia della BAM, contenuta nel Directory Header Block, è stata modificata per documentare i nuovi settori allocati per il record. La routine non riscrive la Directory Header aggiornata su disco. Per farlo si utilizza PutDirHead

Distrugge: a, y, r0 - r9

Sinossi: La routine alloca un'area della memoria su disco per un record. Il record può sovrapporsi a uno preesistente, che viene interamente e automaticamente cancellato.

Attenzione: la routine cancella il record preesistente *prima che il nuovo*

record venga copiato, quindi se sul disco non c'è abbastanza spazio per contenere la nuova versione, il record originale viene perso senza che la nuova versione possa prendere il suo posto. Per ovviare a questo inconveniente, le applicazioni devono preventivamente chiamare la routine CalcBlksFree per assicurarsi che sul disco ci sia spazio sufficiente per contenere il record. La data e l'ora dell'ultima modifica vengono memorizzate su disco solo all'atto della chiusura del file.

ReadRecord

Funzione: Trasferisce il record corrente in memoria.

Indirizzo: $C28C

Chiama: ReadFile

Parametri:

r2	numero massimo di byte che l'applicazione è in grado di ricevere e memorizzare. Se il record eccede lo spazio di memoria messo a disposizione dall'applicazione, si verifica l'errore BFR_OVERFLOW
r7	indirizzo di memoria dal quale iniziare la memorizzazione del record

La routine OpenRecordFile ha preparato le seguenti variabili: usedRecord, curRecord, fileWritten, fileHeader, curDirHead, dirEntryBuf (fileHeader contiene la tavola indice)

Preparazione del drive: Attraverso SetDevice devono essere impostati curDrive e curDevice, e la BAM del disco corrente dev'essere trasferita in memoria attraverso OpenDisk o NewDisk.

Restituisce:

x	codice d'errore da disco
r7	puntatore al byte che segue l'ultimo trasferito in memoria (se il record non è vuoto)
a	flag di record vuoto: 0 = record vuoto
fileTrScTab	tavola degli indirizzi T/S dei blocchi caricati da disco
r5L	puntatore in fileTrScTab all'indirizzo T/S (word) dell'ultimo blocco caricato
r1	se si verifica l'errore BFR_OVERFLOW, r1 contiene l'indirizzo T/S del blocco che non è stato letto perché è stata raggiunta la massima capienza del buffer

Distrugge: a, y, r1 - r4

Sinossi: Il record corrente viene letto e trasferito in memoria nel buffer il cui indirizzo è specificato da r7. La routine aggiorna la tavola fileTrScTab. Le prime due word nella tavola fileTrScTab sono identiche: entrambe contengono l'indirizzo T/S del primo blocco del record.

InsertRecord

Funzione: Nella copia del blocco indice del file VLIR (mantenuta in memoria a fileHeader), InsertRecord inserisce un nuovo record vuoto nella posizione corrente, spostando in avanti tutti quelli che seguono.

Indirizzo: $C286

Parametri:

curRecord individua il record alla cui posizione dev'essere inserito un nuovo record vuoto

fileHeader deve contenere la copia del blocco indice del file VLIR

Restituisce: Il nuovo record alla posizione individuata da curRecord, e tutti i record che seguono spostati di una posizione in avanti

Distrugge: a, x, y, r0L

Sinossi: Nella copia del blocco indice del file VLIR (mantenuta in memoria a fileHeader), InsertRecord, a partire dal record individuato da curRecord compreso, sposta in avanti tutti i puntatori T/S dei record successivi di uno spazio puntatore (2 byte), creando così lo spazio per il puntatore a un nuovo record. Questo spazio viene generato in corrispondenza di curRecord, e viene aggiornato con i valori $00 e $FF per indicare che il record è vuoto.

Il blocco indice conservato in memoria non viene riscritto su disco.

AppendRecord

Funzione: Dopo aver ricevuto tra i parametri la posizione di curRecord, la routine crea uno spazio puntatore nella posizione immediatamente successiva.

Indirizzo: $C289

Parametri:

curRecord	individua il record dopo il quale dev'essere inserito un nuovo record vuoto
fileHeader	deve contenere la copia del blocco indice del file VLIR

Restituisce: Il nuovo record nella posizione successiva a quella corrente
curRecord punta al record appena inserito

Distrugge: a, x, y, r0L

Sinossi: Dopo aver ricevuto curRecord, la routine crea uno spazio puntatore nella posizione immediatamente successiva. Per creare lo spazio necessario, sposta in avanti tutti i record da curRecord+1 in poi. L'operazione viene eseguita nel buffer fileHeader che dovrebbe contenere la copia del blocco indice del file VLIR. In pratica, l'unica differenza con la precedente routine è che InsertRecord utilizza curRecord per puntare la posizione in cui viene inserito il nuovo record vuoto, mentre AppendRecord si serve del puntatore per individuare il record dopo il quale deve inserire il nuovo record vuoto.

Dopo l'esecuzione della routine, curRecord punta al record vuoto appena inserito.

18 I DRIVER DI STAMPA

Questo capitolo si rivolge in modo particolare ai programmatori che:

1) desiderano utilizzare i driver di stampa GEOS nell'ambito delle loro applicazioni
2) desiderano realizzare un driver di stampa GEOS in grado di comandare una stampante non ancora prevista da GEOS.

L'attuale situazione delle stampanti

Oggi sul mercato è presente una tale quantità di stampanti di ogni tipo che si dovrebbero scrivere ancora molti libri, oltre a quelli già pubblicati, per descriverne tutte le caratteristiche. Per ottenere le informazioni necessarie al funzionamento di una specifica stampante o di un'interfaccia, si deve consultare il manuale allegato in dotazione o rivolgersi al rivenditore ufficiale.

Esistono in commercio due grandi categorie di stampanti: quelle "a caratteri" (per esempio, le stampanti a margherita o a globo rotante) e quelle "a matrice di punti". Le stampanti a caratteri utilizzano per la stampa caratteri solidi predefiniti. Quelle a margherita, per esempio, permettono di cambiare la margherita di caratteri per disporre di una fonte diversa, ma non offrono certo la possibilità di stampare qualsiasi tipo di carattere o forma geometrica. La loro maggior limitazione sta nell'impossibilità da parte del computer di definire via software il tipo di carattere. Non sono in grado, per esempio, di stampare una pagina in bit-map. Dal momento che GEOS lavora quasi esclusivamente in alta risoluzione, tanto per i disegni quanto per i testi, le stampanti a caratteri non si prestano a essere utilizzate in ambiente GEOS. Per come è stato progettato e per l'uso a cui è destinato, GEOS richiede l'impiego di una stampante a matrice di punti.

Analizziamo il funzionamento di una stampante a matrice di punti. La testina di stampa è costituita da una linea verticale di punti, ciascuno dei quali può lasciare la sua impronta sulla carta. In una stampante ad aghi, la linea verticale è costituita da una serie di aghi che premono un nastro inchiostrato, generando sulla carta la stampa di un punto. In una stampante a getto, i punti sono generati invece da piccoli spruzzi d'inchiostro sulla carta. In entrambi i casi, il principio è lo stesso: la stampante lavora attivando opportunamente i punti della matrice monodimensionale verticale, ed è così in grado di ottenere qualunque tipo di disegno per i caratteri. La riga di stampa si ottiene con il movimento orizzontale della testina. Il software di gestione, che sia quello interno della stampante o che risieda nel computer, deve passare alla testina di stampa il disegno da riprodurre per ogni spazio carattere orizzontale. A ogni posizione orizzontale lungo la riga di stampa, la testina riproduce sulla carta i disegni ricevuti. Questi disegni possono venire da uno dei set di caratteri memorizzati su ROM all'interno della stampante, o possono venire direttamente dal computer. Nel primo caso, la stampante richiede solo che il computer specifichi il codice ASCII del carattere, ottenuto il quale accede alla propria ROM caratteri per prelevare il disegno e passarlo alla testina di stampa. In questo modo il computer può far stampare solo i caratteri della fonte di cui è dotata la stampante. Nel secondo caso il computer ha il completo controllo del disegno da passare alla testina, e può quindi ottenere la stampa di qualunque figura. GEOS adotta questo secondo sistema. In realtà la maggior parte delle stampanti a matrice di punti ricevono il disegno da stampare come una matrice bidimensionale di pixel, ma la testina di stampa è composta da una sola linea verticale di punti che si muove orizzontalmente per creare un disegno bidimensionale. Nella figura che segue appare la linea verticale di punti della testina e due esempi di stampa. La testina si muove orizzontalmente generando la matrice di punti bidimensionale che costituisce il carattere.

La generazione della matrice grafica bidimensionale

Testina	**Caratteri stampati**	
0	---0---	-0-----
0	--0-0--	00-----
0	-0---0-	-0-----
0	0000000	-0-----
0	0-----0	-0-----
0	0-----0	-0-----
0	0-----0	000----

La stampa in caratteri ASCII e quella in modo grafico

Convenzionalmente le stampanti a matrice di punti si definiscono "grafiche" se oltre a stampare i codici ASCII che ricevono, sono anche in grado di stampare caratteri definiti direttamente dal computer. Nel caso dei codici ASCII, il computer riempie il buffer della stampante con i codici ASCII da stampare, e la stampante preleva il disegno di ogni carattere dal proprio set interno, dove sono memorizzate tutte le matrici di punti associate a ciascun carattere stampabile. Nel secondo caso, il computer ha la facoltà di passare alla stampante la matrice di punti volta per volta, riuscendo così a stampare qualsiasi forma grafica.

Il set di caratteri interno della stampante viene utilizzato sia nella stampa normale sia in quella "ad alta qualità" (near letter quality o NLQ). Si parla di stampa normale quando l'applicazione passa al driver di input una stringa di normali caratteri ASCII (non Commodore ASCII) da stampare, e la periferica li stampa con una sola passata orizzontale alla massima velocità, prelevando le matrici di punti dal set di caratteri interno memorizzato su ROM. In modo NLQ la stampante accede ancora alla ROM interna per prelevare le matrici di punti, ma la testina compie più di un passaggio sulla stessa riga per ottenere una migliore qualità di stampa (non si tratta comunque dell'unico metodo disponibile per ottenere la stampa in modo NLQ).

GEOS normalmente seleziona il modo grafico per la stampa dei disegni e dei testi, in modo da poter stampare qualsiasi tipo di fonte carattere. A seconda del tipo di stampante, il modo grafico può essere chiamato Graphics Mode, Bit-Image Mode o APA Graphics Mode (All Point Addressable, punti tutti indirizzabili). Con quest'ultima tecnica la stampante interpreta i byte memorizzati nel buffer non come codici ASCII, ma come matrici monodimensionali verticali da passare alla testina di stampa. La tavola che riportiamo nella pagina seguente mostra un esempio di come una tipica testina può essere indirizzata nel modo grafico. A ogni punto della linea verticale è assegnato un bit. Nella tavola compaiono i valori esadecimali contenuti nel buffer di stampa e il loro effetto sui punti della testina.

Rappresentazione dei dati grafici su carta

Bit assegnato al punto	**Punti della testina**	**Valori dei byte nel buffer e immagine riprodotta su carta**										
		$01	$02	$04	$08	$10	$20	$40	$80	$AA	$55	$00
0	0	0	-	-	-	-	-	-	-	-	0	-
1	0	-	0	-	-	-	-	-	-	0	-	-
2	0	-	-	0	-	-	-	-	-	-	0	-
3	0	-	-	-	0	-	-	-	-	0	-	-
4	0	-	-	-	-	0	-	-	-	-	0	-
5	0	-	-	-	-	-	0	-	-	0	-	-
6	0	-	-	-	-	-	-	0	-	-	0	-
7	0	-	-	-	-	-	-	-	0	0	-	-

Stampanti a matrice di punti

Sono presenti in commercio principalmente due categorie di testine a matrice di punti: quelle a 9 e quelle a 24 punti. Le stampanti dotate di testine a 9 punti, quando funzionano in modo grafico, indirizzano esclusivamente i punti più in alto, 7 o 8 punti a seconda del tipo di stampante. I punti più in basso sono di solito impiegati solo per creare i caratteri discendenti come la "g" e la "p". Le stampanti a matrice di punti si possono differenziare anche sotto altri aspetti. Per esempio alcune possono assegnare il bit 0 del byte al primo punto in alto della testina, mentre altre possono assegnarlo all'ultimo punto in basso. Il driver di stampa deve tenere conto delle particolari caratteristiche della stampante che comanda, togliendo all'applicazione il problema di doversi preoccupare anche delle differenze tra una stampante e l'altra. Dal momento che il computer gestisce i dati a 8 bit più facilmente di quelli a 7 bit, la gestione delle stampanti dotate di testina a 7 punti verticali (modo grafico) può presentare alcuni problemi. Analizzeremo questi problemi quando introdurremo gli algoritmi di stampa. Per il momento ci limitiamo a una descrizione generale.

Nella maggior parte dei casi, viene stampato un punto di altezza e larghezza pari a 1/72 di pollice, separato dal punto contiguo lungo la verticale ancora di 1/72 di pollice. Le colonne di punti hanno di solito una distanza orizzontale pari a 1/60, 1/72, 1/80 di pollice, o anche meno, a seconda del tipo di stampante e del modo di stampa selezionato.

Le stampanti con testina a 24 punti hanno fondamentalmente lo stesso funzionamento delle stampanti con testina a 9 punti. Possiedono ovviamente una

miglior risoluzione verticale (verticalmente ci sono 24 punti nello spazio che prima era occupata da soli 9 punti), mentre la risoluzione orizzontale rimane invariata.

L'attivazione del modo grafico, e la conseguente disattivazione, avviene fondamentalmente in due modi. Alcune stampanti, una volta ricevuto il comando di attivazione del modo grafico, rimangono in questo stato sino a quando non ricevono il comando di disattivazione. Altre stampanti richiedono l'invio del comando di attivazione del modo grafico seguito da un byte di conteggio; il modo grafico rimane attivo sino a quando la stampante non riceve il numero di byte specificato dal byte di conteggio.

Quando la stampante cede all'applicazione il controllo dei punti della testina, quest'ultima deve indicare alla stampante anche quanto spazio dev'essere lasciato fra una linea e la successiva. Fortunatamente, le stampanti grafiche sono in grado di ricevere dal computer un comando di avanzamento della carta quando incontrano il carattere LF (Line Feed). Per parlare in modo un po' più approfondito dei diversi metodi di stampa, è necessario innanzi tutto imparare a comunicare con la stampante.

Le comunicazioni con la stampante

Questo paragrafo descrive i compiti del bus seriale, e le routine disponibili nel Kernel del C-64 per la comunicazione con i dispositivi periferici.

In genere il C-64 comunica con le sue periferiche (disk drive, stampanti...) attraverso il bus seriale. Questo bus è in grado di collegarsi a cinque dispositivi esterni simultaneamente. Il C-64 è il controllore del bus seriale e può impartire alle periferiche collegate tre tipi fondamentali di comandi: "control", "talk" e "listen" (avviso di controllo, trasmetti, ricevi). Direttamente tramite il bus seriale, il C-64 può impartire alla periferica il comando "talk" (la periferica deve iniziare a trasmettere dati attraverso il bus), "listen" (la periferica deve porsi in ascolto per ricevere dati). Ogni dispositivo esterno è identificabile attraverso il suo numero o indirizzo di device (alcuni dispositivi permettono, con controlli hardware e software, che sia cambiato il loro numero di device). Quando il C-64 trasmette a tutte le periferiche collegate sul bus il comando "control", esse rispondono trasmettendo il loro numero di device, cioè un byte che identifica il tipo di dispositivo. Nella maggior parte dei casi le stampanti a interfaccia seriale trasmettono sul bus il numero di device 4. Quando il C-64 seleziona il dispositivo che deve porsi in ascolto, tale dispositivo, per esempio la stampante, dev'essere in grado di riconoscere sul bus il proprio numero di device (4) e prelevare il byte successivo, che costituisce il comando vero e proprio. E naturalmente dev'essere in grado di riconoscere come un comando il secondo byte, che di solito viene chiamato "indirizzo secondario". Per ottenere informazioni più precise sul funzionamento del bus seriale, consultare il volume *Commodore 64, Guida di riferimento per il programmatore* (Commodore Italiana).

Il Kernel del C-64 contiene alcune routine destinate specificamente alle operazioni con il bus seriale. Queste routine, rispettivamente "talk", "un-talk", "listen" e

"un-listen", trasmettono l'indirizzo secondario e quindi ricevono o trasmettono dati attraverso il bus seriale. In genere, all'atto della chiamata di una di queste routine, l'accumulatore deve contenere il numero di device per identificare la periferica. Quando terminano la loro esecuzione restituiscono nell'accumulatore lo stato dell'operazione effettuata (codice d'errore). Le routine del Kernel impostano a 1 il flag carry del PSW per indicare che il valore contenuto nell'accumulatore è un codice d'errore e non un numero casuale. I driver di stampa utilizzano queste routine primitive per attivare la trasmissione dei dati alla stampante attraverso la porta seriale. Per ottenere ulteriori informazioni sulle routine del Kernel del C-64, si veda la *Commodore 64, Guida di riferimento per il programmatore* (Commodore Italiana).

Particolari sulle interfacce parallele

Dal momento che la maggior parte delle stampanti di buona qualità non sono predisposte per essere collegate direttamente al bus seriale del C-64, anche perché molte utilizzano interfacce parallele Centronics, occorre un'interfaccia seriale-parallela in grado di convertire il protocollo di trasmissione utilizzato dal bus del C-64 nel protocollo Centronics e viceversa. Questi dispositivi sono da tempo disponibili sul mercato, e si possono rintracciare nei migliori negozi specializzati.

I driver di stampa GEOS

Ora che abbiamo analizzato gli aspetti più generali della comunicazione stampante-computer, possiamo procedere alla descrizione dei driver di stampa. Vi sono due esigenze fondamentali che devono essere soddisfatte perché una stampante sia in grado di dialogare con i diversi tipi di applicazione:

1) tutte le applicazioni devono comunicare con il driver di stampa nello stesso modo
2) il driver di stampa deve adattarsi alla stampante in maniera trasparente per l'applicazione.

L'applicazione e il driver di stampa si dividono equamente il lavoro necessario per comunicare con la stampante.

L'interfaccia per la stampa grafica

I driver di stampa comunicano con le applicazioni tramite un buffer da 640 byte per il trasferimento dei dati. Questo buffer, chiamato buffer utente, è in grado di contenere

8 linee di scansione da 80 byte per linea (larghezza massima che GEOS è in grado di gestire), cioè 80 matrici grafiche. Alcune applicazioni prevedono pagine di larghezza più contenuta. Per esempio, in geoWrite V1.3 una riga non può superare i 60 byte. In questo caso l'applicazione deve preoccuparsi di inserire nel buffer un numero opportuno di byte azzerati su entrambi i lati di ogni linea di scansione in modo che la parte significativa di ognuna venga ben centrata sul foglio.

L'applicazione deve preparare un buffer di dati grafici organizzati in modo bit-map. Quando vengono memorizzati nel buffer, questi dati devono essere organizzati nello stesso formato non compatto che possiedono nelle schermate in alta risoluzione, e quindi per prima cosa è necessario espanderli. Il buffer contiene una riga della mappa grafica alta otto linee di scansione (riga grafica). Una volta che il buffer è pronto, l'applicazione deve chiamare la routine del driver di stampa appositamente predisposta per leggerlo, codificarlo e trasmetterlo alla stampante attraverso il bus seriale. Quello che serve al programmatore, quindi, è sapere qual è il formato di organizzazione del buffer e quali routine ha a sua disposizione per comunicare con la stampante. Chi vuole realizzare un driver di stampa deve predisporre la serie di routine standard a cui tutte le applicazioni sono uniformate. Solo così l'applicazione può essere completamente indifferente al tipo di driver di stampa. È sufficiente che il driver possieda le routine standard e che gli entry point (punti di ingresso) di queste routine non varino da driver a driver. Ogni byte memorizzato nel buffer rappresenta 8 pixel orizzontali della mappa grafica. La stampante però si aspetta che il byte in arrivo rappresenti 8 pixel verticali, ed è quindi necessario che le routine del driver di stampa tengano conto di questa incompatibilità e provvedano alle dovute elaborazioni. La stampa di una linea grafica alta 8 pixel avviene utilizzando le seguenti routine.

GetDimensions: Restituisce le dimensioni della pagina grafica che la stampante è in grado di gestire. Queste dimensioni sono espresse in spazi carattere Commodore (le dimensioni di un carattere sullo schermo del C-64).

InitForPrint: Questa routine dev'essere eseguita per inizializzare la stampante prima di procedere alla stampa di un documento. Attualmente si esegue solo per impostare la velocità di trasmissione (baud rate).

StartPrint: Inizializza il bus seriale prima della stampa di ogni pagina e apre un file logico di comunicazione con la stampante, in modo che si possano utilizzare le routine del Kernel del C-64 per trasferire i dati.

PrintBuffer: Stampa in modo grafico il contenuto del buffer utente da 640 byte preparato dall'applicazione.

StopPrint: Gestisce le operazioni da effettuare quando viene determinato lo stato di fine pagina. Trasmette alla stampante il comando form feed (avanza a nuova pagina), e cancella dal buffer le linee di scansione che rimangono nel caso che la stampante stampi solo 7 bit di ogni byte che riceve, avendo cura però di averle già stampate.

L'applicazione ha il completo controllo della stampante. Essa chiama InitForPrint una sola volta per inizializzare la stampante. Per attivare il bus seriale deve eseguire la routine StartPrint. Prima di procedere alla stampa del documento, l'applicazione deve chiamare la routine GetDimensions per sapere qual è la larghezza massima consentita dalla stampante e il numero massimo di linee in una pagina. A questo punto può riempire il buffer utente con i dati della parte di mappa grafica da stampare e chiamare PrintBuffer per procedere alla stampa del buffer. Come già sottolineato, la parte di mappa grafica che si trasferisce nel buffer, cioè la riga grafica, deve avere lo stesso formato con cui viene memorizzata quando si deve visualizzarla sullo schermo ad alta risoluzione. L'applicazione continua questa procedura di aggiornamento del buffer e di successiva stampa fino a quando non ha stampato l'intera pagina. A questo punto dev'essere eseguita la routine StopPrint per far avanzare la carta della stampante fino alla pagina successiva ed eseguire le operazioni necessarie per la chiusura della pagina. Se l'applicazione deve stampare un'altra pagina, deve ripetere tutta la procedura iniziando di nuovo con una chiamata a StartPrint.

La stampa dei caratteri ASCII

La pagina di un documento stampato in caratteri ASCII è convenzionalmente larga 80 caratteri e alta 66 linee. L'applicazione deve trasmettere al driver di stampa una stringa di caratteri ASCII a terminazione nulla. Qualsiasi formattazione e giustificazione del testo, come l'aggiunta di spazi per inserire i tabulatori, dev'essere svolta dall'applicazione. Ogni *end of line* (fine della riga) dev'essere segnalata al driver di stampa attraverso il carattere CR (ritorno carrello). Il driver trasmette quindi alla stampante il comando line feed, che muove la testina all'inizio della riga successiva. Per alcune applicazioni (geoPaint, per esempio), non ha senso stampare documenti in caratteri o in NLQ (near letter quality). Per altre, invece, la cosa è possibile; ma in quest'ultimo caso gli eventuali caratteri di controllo presenti nel testo non devono essere trasmessi alla stampante.

La procedura per la stampa in ASCII si avvicina molto a quella per la stampa grafica. Innanzi tutto l'applicazione deve chiamare InitForPrint per inizializzare la stampante. Quindi eseguire la routine SetNLQ per attivare il modo NLQ di stampa, e infine chiamare StartASCII per preparare il bus seriale (anziché StartPrint, come dovrebbe fare nel caso di stampa grafica). Solo allora l'applicazione può iniziare la vera e propria

trasmissione del testo al driver di stampa. Il testo è una stringa di caratteri a terminazione nulla puntata da r0. Al suo interno le singole linee devono essere separate da caratteri CR, e devono contenere gli spazi necessari per l'impaginazione.

StartASCII: Utilizzata per la stampa in modo caratteri o NLQ. Per il resto è identica a StartPrint.

PrintASCII: Fa le veci di PrintBuffer nel corso della stampa in modo caratteri o NLQ. L'applicazione non passa più al driver il buffer da 640 byte, ma una stringa di caratteri ASCII a terminazione nulla, e la stampante la visualizza su carta utilizzando il proprio set di caratteri.

SetNLQ: Trasmette alla stampante la particolare stringa comando che attiva il modo NLQ.

Le chiamate al driver da parte dell'applicazione

I driver di stampa sono assemblati a partire dall'indirizzo PRINTBASE ($7900), e possono estendersi sino all'indirizzo $7F3F. Le applicazioni devono lasciare questo spazio di memoria libero per i driver. Al di fuori di quest'area, le applicazioni devono allocare spazio per due buffer: il primo viene utilizzato per passare al driver le 80 matrici (640 byte) che compongono la linea grafica da 8 pixel in altezza e 80 byte in larghezza, il secondo (da 1920 byte) viene impiegato dal driver per le sue procedure interne. Questo secondo buffer, il buffer di stampa, è di dimensioni maggiori rispetto al primo in quanto alcuni driver di stampa, per migliorare l'efficienza o per comandare particolari stampanti, richiedono 1920 byte liberi per le loro procedure interne. Le applicazioni, per essere compatibili con tutti i driver possibili, devono rendere disponibili 1920 byte per il buffer di stampa. Quando l'applicazione chiama una routine del driver, i due buffer devono essere puntati rispettivamente da r0 e r1.

Ogni driver di stampa deve contenere, nella sua parte iniziale, una piccola jump table con gli indirizzi delle routine principali ad esso riservate. In questo modo le applicazioni possono chiamare le routine di stampa attraverso i punti d'ingresso presenti nella jump table, indipendentemente dal tipo di driver.

Jump table del driver di stampa

```
                .psect          PRINTBASE

InitForPrint:   jmp             r_InitForPrint   ;prima routine a PRINTBASE
StartPrint:     jmp             r_StartPrint     ;seconda routine a PRINTBASE + 3
PrintBuffer:    jmp             r_PrintBuffer    ;terza routine a PRINTBASE + 6
StopPrint:      jmp             r_StopPrint      ;quarta routine a PRINTBASE + 9
GetDimensions:  jmp             r_GetDimensions  ;quinta routine a PRINTBASE + 12
PrintASCII:     jmp             r_PrintASCII     ;sesta routine a PRINTBASE + 15
StartASCII:     jmp             r_StartASCII     ;settima routine a PRINTBASE + 18
SetNLQ:         jmp             r_SetNLQ         ;ottava routine a PRINTBASE + 21
```

La gestione del driver di stampa da parte dell'applicazione

Ecco le procedure che le applicazioni devono seguire per effettuare la stampa di un documento in modo grafico o in modo caratteri ASCII.

Per la stampa grafica:

A. Si chiama GetDimensions (PRINTBASE + 12) per ottenere le seguenti informazioni:
   1) lunghezza massima della linea di stampa (individuata dalla costante CARDSWIDE), in genere 80 o 60. Questo valore viene restituito in x
   2) numero di righe in una pagina (individuato dalla costante CARDSDEEP), che coincide con il numero delle chiamate a PrintBuffer. Questo valore viene restituito in y.

B. Si chiama InitForPrint (PRINTBASE), a ogni nuovo documento, per inizializzare la stampante. Si chiama StartPrint (PRINTBASE + 3), a ogni nuova pagina, per aprire il file Commodore di output sul bus seriale. Se si verifica un errore, il carry viene impostato a 1 e x contiene il codice dell'errore. In caso contrario x contiene il valore 0.

C. Per stampare una linea si devono eseguire i passi seguenti:
1) si memorizza la riga grafica (80 matrici) in un buffer da 640 byte e si aggiorna r0 con l'indirizzo del buffer
2) si aggiorna r1 con l'indirizzo del secondo buffer da 1920 byte, che il driver utilizza internamente. Si aggiorna r2 con l'indirizzo della memoria colore per la riga. La testina delle stampanti a colori deve scandire la stessa riga più volte. Ogni scansione viene fatta in un colore diverso e con dati grafici diversi
3) si chiama la routine PrintBuffer (PRINTBASE + 9). Nota: l'indirizzo puntato da r1 dev'essere lo stesso durante tutta la stampa del documento e deve quindi rimanere inalterato fra una chiamata a PrintBuffer e la successiva. Il contenuto di r0 e di r2 può invece cambiare a ogni chiamata. Eseguita quest'ultima operazione, si riprende dal punto 1 fino a quando la pagina non è stata interamente stampata.

D. Alla fine di ogni pagina si chiama la routine StopPrint (PRINTBASE + 9), per "pulire" il buffer di stampa e chiudere il file Commodore di output. Se la stampante ha la testina da 7 punti verticali, le linee di scansione che restano nel buffer puntato da r1 devono essere stampate, e non conservate per la successiva riga di dati. Questa routine trascina la carta fino all'inizio del nuovo foglio.

Nota: CARDSWIDE è il numero di matrici grafiche in alta risoluzione utilizzate dalla stampante su una stessa linea. Ogni matrice grafica misura 8 x 8 pixel e quindi ha le dimensioni di uno spazio carattere.
CARDSDEEP è il numero di righe grafiche in alta risoluzione utilizzate dalla stampante in un foglio.

Per la stampa in caratteri ASCII:

A. Si chiama InitForPrint (PRINTBASE) a ogni nuova pagina, per inizializzare la stampante.

B. Si chiama SetNLQ (PRINTBASE + 21) se si desidera la stampa in modo near letter quality.

C. Si chiama StartASCII (PRINTBASE + 18) a ogni nuova pagina, per aprire il file Commodore di output. Se si verifica un errore, il carry viene impostato a 1 e x contiene il codice dell'errore. In caso contrario x contiene il valore 0.

D. Per stampare una riga si devono eseguire i passi seguenti:

1) si memorizza in un buffer il testo che si desidera stampare (la stringa di caratteri ASCII a terminazione nulla), e si aggiorna r0 con l'indirizzo del buffer. Al termine di ogni linea dev'essere inserito il carattere CR perché alla stampante venga trasmesso il Line Feed
2) si chiama la routine PrintASCII (PRINTBASE + 15). Eseguita quest'ultima operazione, si riprende dal punto 1 fino a quando la pagina non è stata interamente stampata.

E. Alla fine di ogni pagina si chiama la routine StopPrint (PRINTBASE + 9), perché la stampante trascini la carta fino all'inizio del foglio successivo. Questa routine chiude anche il file Commodore di output.

Iniziamo ora una dettagliata descrizione di tutte le routine sopra citate.

InitForPrint

Funzione: Inizializza la stampante per la stampa di un documento. Il protocollo d'inizializzazione dipende dal tipo di stampante.

Indirizzo: $7900 (PRINTBASE)

Parametri: Nessuno

Restituisce: Niente

Distrugge: Dipende dal tipo di stampante

Sinossi: InitForPrint è la prima routine da eseguire. Ha il compito di trasmettere alla stampante gli eventuali comandi speciali necessari per stampare documenti in formato GEOS. Con alcune stampanti non esegue alcuna operazione.

GetDimensions

Funzione: Restituisce le dimensioni in matrici grafiche del rettangolo che verrà stampato in un'area di 8 x 10,5 pollici della pagina.

Indirizzo: \$790C (PRINTBASE + 12)

Parametri: Nessuno

Restituisce:
- x larghezza della linea, in matrici grafiche
- y altezza del foglio, in righe grafiche (una riga grafica è alta quanto una matrice grafica)

Distrugge: Niente

Sinossi: GetDimensions fornisce informazioni sulle capacità di risoluzione della stampante in un'area di 8 x 10,5 pollici (le dimensioni del foglio di stampa). Alcune stampanti in commercio non possono stampare più di 60 punti per pollice e quindi non sono in grado di stampare l'intera riga grafica di GEOS (640 pixel). In questo caso l'applicazione deve decidere quale parte della pagina stampare e quale sacrificare. geoPaint, per esempio, stampa le prime 60 matrici grafiche partendo da sinistra. geoWrite, invece, salta le prime 10 matrici grafiche, lascia che il driver stampi le 60 matrici successive e taglia le ultime 10. Si stampano cioè solo le 60 matrici grafiche centrali, sebbene il documento ne contenga 80 per riga.

GetDimensions fornisce in x la massima larghezza di linea possibile per la stampante, misurata in matrici grafiche (8 x 8 pixel); in y fornisce il numero che indica quante volte dev'essere riempito il buffer per completare una pagina grafica su carta (cioè il massimo numero di righe per ogni pagina; ogni riga è alta 8 pixel). Dopo aver aggiornato i registri x e y, accedendo a una tavola di valori memorizzata all'inizio dei codici del driver di stampa, GetDimensions ha finito il suo compito.

Quando viene selezionato il modo a caratteri o NLQ, non è necessario chiamare GetDimensions. GEOS assume che ogni documento in caratteri ASCII debba essere stampato su fogli da 66 righe con 80 caratteri per riga.

StartPrint

Funzione: Inizializza il bus seriale per comunicare con la stampante. Prepara la stampante alla ricezione di dati grafici.

Indirizzo: $7903 (PRINTBASE + 3)

Parametri: Nessuno

Restituisce: x codice dell'eventuale errore

Distrugge: Dipende dal tipo di stampante

Sinossi: StartPrint serve ad aprire una struttura di file Commodore fittizia. Tramite questa struttura vengono trasmessi i dati alla stampante sul bus seriale. La struttura di file Commodore non viene realmente creata, ma si fa in modo che la stampante creda di ricevere i dati da un file. Vediamo come vengono utilizzate le routine del Kernel del C-64 per effettuare la comunicazione. Alla stampante è associato il numero di dispositivo 4. StartPrint chiama SetDevice per fare in modo che GEOS indirizzi la stampante. Quindi chiama InitForIO in modo che vengano disabilitati gli interrupt e attivati lo spazio di I/O e la ROM del Kernel del C-64. A questo punto StartPrint utilizza le primitive del Kernel del C-64 per simulare l'apertura di un file di comunicazione con la stampante. Le primitive del Kernel di GEOS restituiscono lo stato delle operazioni nella locazione $90. Se StartPrint vede che la locazione $90 non è azzerata, trasferisce il valore in essa contenuto nel registro x, chiude il file e restituisce il controllo all'applicazione. In questo modo il codice d'errore restituito in x corrisponde al numero restituito dalle primitive del Kernel del C-64. L'applicazione dev'essere in grado di gestire i due errori "Device Not Present" e "I/O Timeout Errors". Se invece il file è stato aperto con successo, la stampante riceve il comando di "mettersi in ascolto" sul bus seriale e gli altri comandi introduttivi di cui ha bisogno. Come ultima operazione, StartPrint ordina di cessare l'ascolto sul bus seriale e di eseguire la routine DoneWithIO per ripristinare la configurazione standard di sistema. A questo punto StartPrint restituisce il controllo all'applicazione, memorizzando nel registro x il valore 0 per indicare che non sono avvenuti errori durante lo svolgimento delle operazioni. Attenzione: se StartPrint ha avuto successo, la stampante si comporta come se fosse stato aperto un file Commodore, dal quale dovrebbe ricevere i dati.

PrintBuffer

Funzione: Manda in esecuzione la stampa dell'intero buffer da 640 byte (80 matrici grafiche) contenente i dati grafici.

Indirizzo: $7906 (PRINTBASE + 6)

Parametri: r0 indirizzo del buffer da 640 byte (80 matrici grafiche) che dev'essere stampato

r1 indirizzo del buffer di stampa da 1920 byte che il driver utilizza internamente

Nota: l'indirizzo di quest'ultimo buffer non deve cambiare fra una chiamata a PrintBuffer e la successiva. I driver di stampa per le stampanti a 7 bit utilizzano questo buffer per non perdere le linee di scansione che restano dopo ogni chiamata di PrintBuffer. Infatti, a ogni chiamata di PrintBuffer vengono passate 8 linee di scansione, ma soltanto 7 vengono stampate

r2 indirizzo degli 80 byte della memoria colore per la matrice della riga

Restituisce: r0, r1 inalterati

Distrugge: Tutti i registri

Sinossi: PrintBuffer viene eseguita per stampare la linea grafica in alta risoluzione contenuta nel buffer puntato da r0 (il buffer utente). Una linea grafica è composta da 80 matrici grafiche (8 x 8 pixel). La routine esegue nuovamente le chiamate alle routine SetDevice e InitForIO prima di ordinare alla stampante di "mettersi in ascolto". Se la testina di stampa è in grado di stampare 8 punti contemporaneamente, PrintBuffer trasmette semplicemente il contenuto del buffer utente puntato da r0.

Se invece la testina è in grado di stampare solo 7 punti contemporaneamente, le operazioni da compiere sono più complesse dal momento che l'ultima linea di dati non può essere stampata. La prima chiamata a PrintBuffer serve per la stampa delle prime 7 linee di scansione contenute nel buffer. L'ultima linea viene tenuta per la prossima riga grafica. Alla seconda esecuzione di PrintBuffer, vengono stampate la linea rimasta e 6 linee prelevate dal buffer utente, lasciando due linee nel buffer di stampa. Il

processo di stampa continua nello stesso modo finché le linee conservate nel buffer di stampa diventano 8, cioè sono state stampate 7*8 = 56 linee di scansione. Ora la routine deve stamparle come una normale riga grafica, che però non è stata passata dall'applicazione, ma rappresenta un accumulo di dati creato da PrintBuffer. Stampando questa eccedenza di 8 linee di scansione, viene prodotto un nuovo accumulo di una linea grafica. Quindi il ciclo riprende dalla condizione iniziale, ma con un riporto composto da una linea di scansione.

Quindi per compiere il necessario riallineamento con le linee grafiche in input ogni 56 linee di scansione stampate (8 righe grafiche * 7 linee stampate), PrintBuffer stampa la riga grafica accumulata prima di prendere in considerazione la nuova riga grafica in input. Il ciclo a questo punto riprende, ma non dall'inizio. Infatti ora alla nuova riga grafica ricevuta dall'utente si aggiunge il riporto della linea di scansione rimasta dopo l'ultimo svuotamento del buffer di stampa. L'algoritmo di stampa per le stampanti a 7 bit verrà meglio illustrato più avanti.

Dopo aver stampato un'intera riga grafica, PrintBuffer trasmette alla stampante il carattere CR (ritorno carrello), e il carattere LF (avanza alla riga successiva) se la stampante lo richiede. In seguito trasmette il comando di terminare l'ascolto sul bus e chiama la routine DoneWithIO per riportare il sistema allo stato precedente alla chiamata. Per ottenere maggiori dettagli si vedano i due esempi di driver di stampa nei prossimi paragrafi.

StopPrint

Funzione: Questa routine dev'essere eseguita dopo la stampa di ogni pagina, per cancellare il buffer di output e comandare alla stampante di trascinare la carta a pagina nuova.

Indirizzo: $7909 (PRINTBASE + 9)

Parametri:

r0 indirizzo del buffer da 640 byte (80 matrici grafiche) utilizzato per passare le righe grafiche al driver

r1 indirizzo del buffer di stampa da 1920 byte che il driver utilizza internamente

Nota: l'indirizzo di quest'ultimo buffer non deve cambiare fra una chiamata a PrintBuffer e la successiva. I driver di stampa per le stampanti a 7 bit utilizzano questo buffer per non perdere le linee di scansione accumulate dopo ogni chiamata di PrintBuffer. In questo caso, a ogni chiamata di PrintBuffer vengono passate alla stampante 8 linee di scansione, ma solo 7 vengono stampate

Restituisce: r0, r1 inalterato

Distrugge: Tutti i registri

Sinossi: StopPrint si esegue quando tutte le righe grafiche che compongono una pagina sono state stampate. Esegue le seguenti operazioni: chiama SetDevice, InitForIO, e ordina alla stampante di mettersi in ascolto sul bus seriale. Se la stampante ha la testina da 7 punti, la routine svuota il buffer di stampa stampando le eventuali linee di scansione accumulate. Quindi avanza la carta alla nuova pagina, comanda alla stampante di cessare l'ascolto sul bus, chiude il file Commodore di output fittizio ed esegue la routine DoneWithIO.

StartASCII

Funzione: Inizializza il bus seriale per comunicare con la stampante. Prepara la stampante alla ricezione di codici ASCII da stampare con il proprio set interno di caratteri.

Indirizzo: $7912 (PRINTBASE + 18)

Parametri: Nessuno

Restituisce: a, x, y, r3

Distrugge: Dipende dal tipo di stampante

Sinossi: StartASCII effettua operazioni molto simili a quelle compiute da StartPrint, cioè apre un file Commodore di output fittizio, in modo che la stampante creda di ricevere i dati da un file logico. Grazie a questo artificio possono essere utilizzate le routine del Kernel del C-64 per le comunicazioni con la stampante. Anziché in modo grafico, la stampante viene inizializzata per ricevere codici ASCII. Per maggiori dettagli si veda la scheda di StartPrint.

PrintASCII

Funzione: Stampa la stringa ASCII a terminazione nulla puntata da r0.

Indirizzo: $790F (PRINTBASE + 15)

Parametri: r0 contiene il puntatore alla stringa ASCII

Restituisce: Niente

Distrugge: Tutti i registri

Sinossi: PrintASCII viene eseguita per stampare un'intera stringa ASCII a terminazione nulla tramite il set di caratteri residente nella stampante. La stringa viene memorizzata nel buffer utente e puntata da r0. La routine ritiene conclusa la stampa della stringa quando incontra il carattere NULL (0). Non ci sono quindi limiti alla lunghezza della stringa. All'interno della stringa possono essere presenti caratteri CR per indicare la fine di una riga e l'inizio della riga seguente. Per alcune stampanti, la routine trasmette anche il carattere LF (avanza riga successiva) quando incontra il carattere CR.

Dopo aver stampato un'intera stringa ASCII, la routine ordina alla stampante di cessare l'ascolto, e chiama DoneWithIO per riportare il sistema alla configurazione precedente.

Il driver per le stampanti a 8 punti

Ecco la descrizione di un driver per stampanti a 8 punti. Per quanto questo driver impieghi esclusivamente il buffer utente da 640 byte, nelle schede viene sempre indicato fra i parametri un buffer di stampa da 1920 byte. Questo accade in quanto l'applicazione non è tenuta a sapere quale driver è stato selezionato, e deve quindi adattarsi alle esigenze generiche di tutti i driver. In linea di principio, il funzionamento di questo driver è molto semplice.

L'applicazione, dopo aver chiamato InitForPrint e StartPrint, deve preparare il buffer utente e cederne l'indirizzo a PrintBuffer. Questa routine apre il canale di comunicazione con la stampante e chiama PrintPrintBuffer. Quest'ultima è il cuore della stampa. Per prima cosa controlla quante matrici grafiche della riga sono azzerate, scorrendo la riga da destra verso sinistra. Questa operazione, eseguita da TestBuffer, permette a PrintPrintBuffer di ottimizzare la stampa della riga grafica evitando di stampare tutta la riga quando solo la parte iniziale contiene matrici grafiche significative.

A questo punto PrintPrintBuffer procede alla stampa delle matrici grafiche significative. Ogni matrice, prima di essere trasmessa alla stampante tramite la routine SendBuff, viene opportunamente ruotata in modo che ogni byte non corrisponda più a una riga orizzontale, ma a una linea verticale di 8 bit. Questa rotazione è necessaria dal momento che la testina di stampa rappresenta su carta i byte che riceve verticalmente.

Questo è l'algoritmo di stampa che normalmente si impiega per comandare le stampanti a 8 punti. Nel prossimo capitolo verrà illustrato l'algoritmo più complesso per le stampanti a 7 punti.

Questo driver è stato ideato per le stampanti:

EPSON	FX-80, FX-100, RX-80, RX-100, JX-80
PANASONIC	KX-1091, KX-1092, KX-1592, KX-1595

ed è stato sottoposto al test di verifica utilizzando la stampante:

EPSON JX-80.

Nella pagina successiva viene presentato il listato del file sorgente utilizzato per il nostro driver di stampa.

Costanti per la stampante

```
TRANSPARENT     = 5                 ;comando che non effettua alcuna operazione
CARDSWIDE       = 80                ;80 matrici grafiche Commodore di larghezza
CARDSDEEP       = 90                ;90 matrici grafiche Commodore di altezza
SECADD          = TRANSPARENT       ;indirizzo secondario
PRINTADDR       = 4                 ;numero di dispositivo della stampante
PRINTBASE       = $7900             ;indirizzo al quale il driver dev'essere rilocato
ESC             = $1B               ;codice escape della stampante

                                    ;per questi file si veda l'appendice
.include        geosMacros          ;macro per l'Assembler della Berkeley
.include        geosConstants       ;costanti
.include        geosMemoryMap       ;locazioni e variabili nella RAM
.include        geosRoutines        ;jump table per le routine di GEOS

.psect          PRINTBASE           ;indirizzo al quale allocare il file oggetto
                                    ;compilato
```

Jump table residente

```
InitForPrint:   rts                         ;prima routine a PRINTBASE
                nop                         ;per questa stampante la routine non deve
                nop                         ;compiere nessuna operazione
StartPrint:     jmp     r_StartPrint        ;seconda routine a PRINTBASE + 3
PrintBuffer:    jmp     r_PrintBuffer       ;terza routine a PRINTBASE + 6
StopPrint:      jmp     r_StopPrint         ;quarta routine a PRINTBASE + 9
GetDimensions:  jmp     r_GetDimensions     ;quinta routine a PRINTBASE + 12
PrintASCII:     jmp     r_PrintASCII        ;sesta routine a PRINTBASE + 15
StartASCII:     jmp     r_StartASCII        ;settima routine a PRINTBASE + 18
SetNLQ:         jmp     r_SetNLQ            ;ottava routine a PRINTBASE + 21
```

Variabili in RAM e utility

```
printerName:     .byte "Epson FX-80",0   ;nome della stampante come dovrebbe apparire
                                         ;nel menu di selezione della stampante
prntBlCard:      .byte 0,0,0,0,0,0,0,0 ;matrice grafica stampabile
breakCount:      .byte 0                 ;contatore linee avanzate
sCount:          .byte 0                 ;contatore delle linee all'interno
                                         ;della matrice grafica corrente
cardCount:       .byte 0                 ;contatore delle matrici grafiche azzerate
                                         ;a fine riga
modeFlag:        .byte 0                 ;$00=stampa grafica, $FF=stampa ASCII
.include         geosUtilities           ;file contenente routine di basso livello per la
                                         ;comunicazione con la stampante. Questo file e'
                                         ;riportato alla fine del capitolo
```

Le routine direttamente accessibili dall'applicazione

GetDimensions

Funzione: Restituisce le dimensioni (in matrici grafiche) del disegno che la stampante è in grado di riprodurre sul foglio.

Parametri: Nessuno

Restituisce:
- x larghezza massima (in matrici grafiche) della linea che la stampante è in grado di stampare
- y altezza massima (in matrici grafiche) dello spazio lungo il foglio entro il quale la stampante è in grado di stampare

Distrugge: Niente

Sinossi: GetDimensions informa sul numero di matrici grafiche che la stampante è in grado di stampare nel sotto-rettangolo di 8 x 10,5 pollici. Il sotto-rettangolo è compreso interamente nel foglio di stampa, che normalmente misura 8,5 x 11 pollici.

```
r_GetDimensions:
 ldx        #CARDSWIDE          ;numero di matrici grafiche orizzontali
 ldy        #CARDSDEEP          ;numero di matrici grafiche verticali
 lda        #0                  ;impostato a 0 per i driver grafici
 rts
```

StartPrint

Funzione: Realizza la necessaria inizializzazione della stampante all'inizio di ogni pagina del documento.

Parametri: Nessuno

Restituisce: x ≠ 0 se la stampante non è accessibile

Distrugge: Tutti i registri

Sinossi: Questa routine prepara per la stampa di una pagina il bus seriale e la stampante. La stampante viene configurata opportunamente per ricevere dati grafici.

```
r_StartPrint:
   lda        #0                      ;imposta il modo grafico
   sta        modeFlag
StartIn:
   lda        #PRINTADDR              ;attiva il dispositivo 4
   jsr        SetDevice
   jsr        InitForIO               ;attiva il Kernel del C-64, lo spazio di I/O e
                                      ;disabilita gli interrupt
   lda        #0
   sta        $90                     ;imposta l'indicatore d'errore a nessun errore
   jsr        OpenFile                ;apre il file fittizio di comunicazione
                                      ;con la stampante
   lda        $90                     ;se si e' verificato un errore con il canale
                                      ;di output, esegui
   bne        20$                     ;la routine di gestione degli errori
   jsr        OpenPrint               ;apre il canale con la stampante
   jsr        InitPrinter             ;inizializza la stampante in modo grafico
   jsr        ClosePrint              ;chiude il canale con la stampante
   jsr        Delay                   ;attende per eliminare eventuali problemi di
                                      ;temporizzazione
   jsr        DoneWithIO              ;ripristina lo stato precedente alla chiamata
                                      ;di InitForIO
```

```
   ldx         #0                    ;nessun errore
   rts
20$:
   pha                               ;salva temporaneamente il codice
                                     ;dell'eventuale errore
                                     ;bit 0 impostato a 1: timeout, scrittura
                                     ;bit 7 impostato a 1: dispositivo non presente
   jsr         CloseFile             ;chiude il file
   jsr         DoneWithIO            ;ripristina lo stato precedente alla chiamata
                                     ;di InitForIO
   pla                               ;recupera il codice dell'errore
   tax                               ;e lo trasferisce in x
   rts
Delay:
   ldx         #0
10$:
   ldy         #0
20$:
   dey
   bne         20$
   dex
   bne         10$
   rts
```

StartASCII

Funzione: Inizializza la stampante Epson in modo che tratti i dati trasmessi come codici ASCII.

Parametri: Nessuno

Restituisce: Niente

Distrugge: Tutti i registri

Sinossi: Imposta il flag modeFlag per indicare la selezione del modo ASCII di stampa. Chiama StartIn per procedere all'inizializzazione. Questa routine dev'essere chiamata all'inizio di ogni nuova pagina.

```
r_StartASCII:
 LoadB       modeFlag, #$FF        ;imposta il modo ASCII di stampa
 jmp         StartIn               ;utilizza una parte di StartPrint per procedere
                                   ;alla preparazione della stampante
```

SetNLQ

Funzione: Inizializza la stampante Epson nel modo near letter quality.

Parametri: Nessuno

Restituisce: Niente

Distrugge: Tutti i registri

Sinossi: Trasmette alla stampante la stringa comando necessaria per selezionare il modo NLQ di stampa.

```
r_SetNLQ:
    lda         #PRINTADDR
    jsr         SetDevice            ;attiva il dispositivo associato
    jsr         InitForIO            ;attiva il Kernel del C-64, lo spazio di I/O
                                     ;e disabilita gli interrupt
    jsr         OpenPrint            ;apre il canale con la stampante
    LoadW       r3, #nlqTbl          ;tavola dei byte d'inizializzazione
                                     ;da trasmettere
    lda         #(enlqTbl-nlqTbl)    ;lunghezza della tavola
    jsr         Strout               ;trasmette la tavola alla stampante
    jsr         ClosePrint           ;chiude il canale con la stampante
    jsr         DoneWithIO           ;ripristina lo stato precedente
                                     ;alla chiamata di InitForIO
    rts

nlqTbl:
    .byte       $47, ESC, $45, ESC   ;questi byte, trasmessi da destra verso sinistra,
                                     ;comandano alla stampante di abilitare
                                     ;il modo NLQ
enlqTbl:                             ;fine della stringa comando per attivare
                                     ;il modo NLQ
```

PrintASCII

Funzione: Trasmette alla stampante una stringa ASCII a terminazione nulla.

Parametri: r0 puntatore alla stringa ASCII

Restituisce: Niente

Distrugge: Tutti i registri

Sinossi: Trasmette la stringa ASCII a terminazione nulla puntata da r0. La stringa può contenere caratteri CR per la separazione delle linee. La routine, quando incontra il carattere CR, trasmette di seguito anche il carattere LF.

```
r_PrintASCII:
   lda      #PRINTADDR
   jsr      SetDevice        ;attiva il dispositivo associato
   jsr      InitForIO        ;attiva il Kernel del C-64, lo spazio di I/O
                             ;e disabilita gli interrupt
   jsr      OpenPrint        ;apre il canale con la stampante
10$:
   ldy      #0               ;azzera l'indice per scorrere la stringa ASCII
   lda      (r0), y          ;preleva il carattere
   beq      30$              ;se raggiunge la fine della stringa, termina
   cmp      #CR              ;se e' un CR aggiunge anche un LF
   bne      20$              ;salta se non e' un CR
   jsr      Ciout            ;trasmette il carattere
   lda      #LF              ;trasmette il LF
20$:
   jsr      Ciout
   IncW     r0               ;punta al carattere successivo
   jmp      10$              ;ripete il loop
30$:
   jsr      ClosePrint       ;chiude il canale con la stampante
   jsr      DoneWithIO       ;ripristina lo stato precedente
                             ;alla chiamata di InitForIO
   rts
```

PrintBuffer

Funzione: Trasmette i dati grafici contenuti all'interno del buffer da 640 byte (80 matrici grafiche) creato dall'applicazione.

Parametri: r0 indirizzo del buffer da 640 byte (80 matrici grafiche)
r1 indirizzo del buffer di stampa da 1920 byte che l'applicazione deve mettere a disposizione del driver per suo uso interno

Nota: l'indirizzo di quest'ultimo buffer non deve cambiare fra una chiamata a PrintBuffer e la successiva. I driver di stampa per le stampanti a 7 bit utilizzano questo buffer per non perdere le linee di scansione accumulate a ogni chiamata di PrintBuffer

Restituisce: r0, r1 inalterati

Distrugge: Tutti i registri

Sinossi: PrintBuffer, come è già stato spiegato, è la routine di livello più alto in grado di trasferire il contenuto del buffer da 640 byte dalla memoria del C-64 alla stampante attraverso la porta seriale.

```
r_PrintBuffer:
   lda      #PRINTADDR
   jsr      SetDevice             ;attiva il dispositivo associato
   jsr      InitForIO             ;attiva il Kernel del C-64, lo spazio di I/O
                                  ;e disabilita gli interrupt
   jsr      OpenPrint             ;apre il canale di comunicazione con la stampante
   MoveW    r0, r3
   jsr      PrintPrintBuffer      ;trasmette la riga grafica alta 8 bit memorizzata
                                  ;nel buffer utente
   jsr      Greturn               ;trasmette i caratteri CR e LF
   jsr      ClosePrint            ;chiude il canale di comunicazione
                                  ;con la stampante
   jsr      DoneWithIO            ;ripristina lo stato precedente alla chiamata
                                  ;di InitForIO
   rts
```

StopPrint

Funzione: Viene eseguita dopo la stampa di ogni pagina, per cancellare il buffer di stampa e comandare alla stampante di far avanzare la carta fino al foglio successivo.

Parametri:
r0 indirizzo del buffer da 640 byte (80 matrici grafiche)
r1 indirizzo del buffer di stampa da 1920 byte che l'applicazione deve mettere a disposizione del driver per suo uso interno

Nota: l'indirizzo di quest'ultimo buffer non deve cambiare fra una chiamata a PrintBuffer e la successiva. I driver di stampa per le stampanti a 7 bit utilizzano questo buffer per non perdere le linee di scansione accumulate a ogni chiamata di PrintBuffer

Restituisce: r0, r1 inalterati

Distrugge: In genere tutti i registri

Sinossi: StopPrint dev'essere chiamata quando è finita la stampa di una pagina. Manda in esecuzione SetDevice, InitForIO, mette in ascolto la stampante, e se la testina è a 7 bit stampa le eventuali linee rimaste nel buffer di stampa. Infine la routine effettua l'avanzamento del foglio all'inizio della nuova pagina, comanda alla stampante di cessare l'ascolto sul bus seriale, chiude il file Commodore di output ed esegue la routine DoneWithIO.

```
r_StopPrint:
   lda       #PRINTADDR
   jsr       SetDevice         ;attiva il dispositivo associato
   jsr       InitForIO         ;attiva il Kernel del C-64, lo spazio di I/O
                               ;e disabilita gli interrupt
   jsr       OpenPrint         ;apre il canale con la stampante
   jsr       FormFeed          ;fa avanzare la carta fino al nuovo foglio
   jsr       ClosePrint        ;chiude il canale di stampa
   jsr       CloseFile         ;chiude il file di output
   jsr       DoneWithIO        ;ripristina lo stato precedente alla chiamata
                               ;di InitForIO
   rts
```

Le routine interne del driver

PrintPrintBuffer

Funzione: Trasmette il buffer di stampa puntato da r3.

Chiamato da: PrintBuffer

Parametri: r3 indirizzo del buffer in memoria

Restituisce: Niente

Distrugge: a, x, y, r0 - r15

Sinossi: Controlla se il buffer è vuoto, tramite TestBuffer, prima di procedere alla stampa della riga grafica. Se il buffer non è vuoto, la routine, per ogni matrice grafica prelevata dal buffer, "fa ruotare" i byte e li trasmette alla stampante. TestBuffer comunica in cardCount quante sono le matrici grafiche azzerate a partire dal lato destro della riga grafica. PrintPrintBuffer, valutando opportunamente cardCount, è in grado di ottimizzare il tempo di stampa trasmettendo esclusivamente le matrici grafiche significative di ogni riga.

```
PrintPrintBuffer:
   PushW     r3              ;salva il puntatore al buffer
   jsr       TestBuffer      ;controlla se il buffer e' completamente azzerato
                             ;e aggiorna cardCount
   bcs       5$              ;se il buffer contiene dati inizia il trasferimento
   PopW      r3              ;altrimenti svuota lo stack e termina
   rts
5$:
   jsr       SetGraphics     ;attiva il modo grafico per la riga corrente
   PopW      r3              ;recupera il puntatore alla matrice
                             ;grafica corrente
   lda       #CARDSWIDE      ;carica la larghezza della riga grafica
                             ;supportata dalla stampante (generalmente 80)
```

```
    sec

    sbc         cardCount           ;sottrae il numero di mappe grafiche nulle
                                    ;a partire dal lato destro della riga grafica
    tax                             ;ottenendo l'esatto numero di matrici grafiche
                                    ;significative della riga grafica corrente
10$:
    txa                             ;salva x
    pha
    jsr         Rotate              ;ruota la matrice grafica puntata da r3
    jsr         SendBuff            ;trasmette la matrice grafica ruotata
    AddVW       #8, r3              ;aggiorna r3 in modo che punti alla matrice
                                    ;grafica successiva
    pla                             ;recupera x
    tax
    dex
    bne         10$                 ;se non ha trasmesso tutte le matrici grafiche
                                    ;riesegue l'operazione con la successiva
                                    ;matrice presente nel buffer
    rts
```

TestBuffer

Funzione: Controlla se il buffer contiene dati grafici da stampare. TestBuffer restituisce in cardCount il numero di matrici grafiche azzerate che incontra lungo la riga grafica scorrendola da destra a sinistra. L'informazione permette a PrintPrintBuffer di non perdere tempo a trasmettere le matrici grafiche azzerate che completano la riga grafica.

Chiamata da: PrintPrintBuffer

Parametri: r3 puntatore all'inizio del buffer da controllare

Restituisce: C impostato a 1 se il buffer contiene dei dati, impostato a 0 se il buffer è vuoto

Distrugge: a, x, r3

Sinossi: Controlla i byte contenuti nel buffer per accertarsi che non siano tutti a 0. Il controllo inizia dal lato destro della riga grafica e procede verso sinistra, matrice per matrice. Durante la scansione delle matrici grafiche, la routine mantiene aggiornato il contatore cardCount. Quando incontra un byte diverso da zero, la routine termina la scansione, imposta a 1 il carry ed esce. Quindi cardCount contiene il numero di matrici grafiche completamente azzerate presenti nella riga grafica da destra a sinistra. Se invece la routine non incontra alcun byte diverso da zero, esce impostando il carry a 0.

```
TestBuffer:
 LoadB      cardCount, #0             ;azzera il contatore delle matrici grafiche
                                      ;azzerate
 ldx        #7                        ;assume che la testina di stampa sia da 8 punti
 stx        sCount                    ;aggiorna il contatore di linee di scansione
                                      ;all'interno della matrice grafica
 AddVW      #((CARDSWIDE-1)*8), r3    ;aggiorna r3 in modo che punti al primo byte
                                      ;dell'ultima matrice grafica lungo la riga
 ldx        #CARDSWIDE                ;aggiorna x con il numero di matrici grafiche
                                      ;presenti in una riga
```

```
10$:
 ldy        sCount          ;preleva il contatore di linee all'interno della
                            ;matrice grafica corrente
20$:
 lda        (r3), y         ;controlla un byte
 bne        30$             ;se e' diverso da 0 l'esecuzione
                            ;della routine termina
 dey                        ;punta al precedente byte nella matrice grafica
 bpl        20$             ;se non ha finito la scansione della matrice grafica
                            ;corrente, controlla il byte precedente
 SubVW      #8, r3          ;aggiorna r3 in modo che punti all'inizio
                            ;della matrice grafica precedente
 inc        cardCount       ;incrementa il contatore delle matrici grafiche
                            ;azzerate
 dex                        ;se non sono state controllate tutte le matrici
                            ;grafiche della riga
 bne        10$             ;vai a verificare la precedente
 clc                        ;se la routine termina con questa istruzione,
                            ;significa che il buffer e' vuoto
 rts
30$:
 sec                        ;imposta il carry a 1 per indicare che il buffer
                            ;non e' vuoto
 rts
```

InitPrinter

Funzione: Inizializza la stampante Epson in modo che utilizzi un'interlinea da 8/72 di pollice se viene attivato il modo grafico, o da 6 linee per pollice se viene attivato il modo ASCII.

Chiamata da: StartPrint

Parametri: modeFlag 0 - modo grafico, $FF - modo ASCII

Restituisce:
r3 #initTbl nel modo grafico, #ainitTbl nel modo ASCII
sCount $FF
y 0

Distrugge: a

Sinossi: Questa routine ha a disposizione due diverse stringhe comando, a seconda che venga selezionato il modo di stampa grafico o quello ASCII. L'opportuna stringa viene inviata alla stampante per l'inizializzazione. Per ottenere maggiori dettagli sulla sintassi dei due comandi, si deve consultare il manuale della stampante.

```
InitPrinter:
 bit        modeFlag                ;stabilisce qual e' il modo grafico selezionato
 bmi        10$                     ;salta se e' stato selezionato il modo ASCII
 LoadW      r3, #initTbl            ;indirizzo della stringa comando
 lda        #(einitTbl-initTbl)     ;lunghezza della stringa
 jmp        Strout                  ;trasmette la stringa e termina
10$:
 LoadW      r3, #ainitTbl           ;indirizzo della stringa comando
 lda        #(eainitTbl-ainitTbl)   ;lunghezza della stringa
 jmp        Strout                  ;trasmette la stringa e termina
initTbl:
 .byte      8, "A", ESC             ;trasmette il nuovo interlinea da 8/72"
 .byte      "@", ESC                ;comando di reset per la stampante
einitTbl:
ainitTbl:
 .byte      2, ESC                  ;comanda di stampare 6 linee per pollice
 .byte      "@", ESC                ;comando di reset per la stampante
eainitTbl:
```

SetGraphics

Funzione: Attiva il modo grafico della stampante Epson (640 punti per linea).

Chiamata da: PrintPrintBuffer

Parametri: cardCount — il numero di matrici grafiche non significative (azzerate) contate dalla fine della riga verso sinistra

Restituisce:

r3	l'indirizzo (memorizzato anche nei due byte a wsdgphTbl) della word in memoria contenente il numero di byte che devono essere trasmessi alla stampante per la linea corrente
sCount	$FF
y	0

Distrugge: a

Sinossi: Seleziona il modo grafico della stampante e trasmette il numero di byte che deve ricevere e stampare in alta risoluzione.

```
SetGraphics:
  LoadB     r3H, #0           ;azzera il byte piu' significativo
  MoveB     cardCount, r3L    ;memorizza cardCount nel byte meno significativo
                              ;in questo modo r3 contiene il numero di matrici
                              ;grafiche non significative a fine riga
  asl       r3L               ;moltiplica r3 per 8 in modo da ottenere
                              ;il numero di byte
  rol       r3H               ;occupati dalle cardCount matrici grafiche
  asl       r3L
  rol       r3H
  asl       r3L
  rol       r3H
  sec
```

```
lda         #<(CARDSWIDE*8)        ;preleva il byte basso del numero di byte
                                   ;che compongono la riga grafica
sbc         r3L                    ;sottrae il byte basso del numero di byte
                                   ;da non stampare
sta         wsdgphTbl+1            ;memorizza il byte basso come secondo byte
                                   ;nella stringa comando
lda         #>(CARDSWIDE*8)        ;preleva il byte alto del numero di byte
                                   ;che compongono la riga grafica
sbc         r3H                    ;sottrae il byte alto del numero di byte
                                   ;da non stampare
sta         wsdgphTbl              ;memorizza il byte alto come primo byte
                                   ;nella stringa comando
LoadW       r3, #wsdgphTbl         ;memorizza in r3 l'indirizzo della stringa comando
lda         #(ewsdgphTbl-wsdgphTbl) ;lunghezza della stringa comando
jmp         Strout                 ;trasmette la stringa comando alla stampante

wsdgphTbl:  .byte 2, 128, 4, "*", ESC  ;N1, N2: byte alto e byte basso del numero di byte
                                   ;che la stampante deve stampare in modo grafico
                                   ;4, "*", ESC: comando di attivazione
                                   ;del modo grafico
ewsdgphTbl:
```

SendBuff

Funzione: Trasmette una matrice grafica stampabile attraverso la porta seriale.

Chiamata da: PrintPrintBuffer

Parametri: prntBlCard questo buffer deve contenere la matrice grafica già ruotata

Restituisce: Niente

Distrugge: a, x

Sinossi: Quando una matrice grafica è stata ruotata in modo tale che i bit di ogni byte corrispondano ai punti verticali della testina di stampa, SendBuff invia i dati della matrice alla stampante attraverso il bus seriale.

```
SendBuff:
 ldx       #0                 ;inizializza il contatore
10$:
 txa                          ;salva il contatore
 pha
 lda       prntBlCard, x      ;preleva il byte da trasmettere
 jsr       Ciout              ;trasmette il byte
 pla                          ;recupera il contatore
 tax
 inx                          ;punta al nuovo byte
 cpx       #8                 ;se non sono stati trasmessi tutti i byte
                              ;della matrice
 bne       10$                ;riprende il loop di trasmissione
 rts
```

Greturn

Funzione: Trasmette alla stampante i caratteri LF (riga nuova) e CR (ritorno carrello).

Chiamata da: PrintBuffer

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, e diverse variabili utilizzate da Ciout

Sinossi: Trasmette alla stampante i caratteri LF (riga nuova) e CR (ritorno carrello).

```
Greturn:
 lda        #CR             ;ritorno carrello
 jsr        Ciout           ;lo trasmette
 lda        #LF             ;riga successiva
 jsr        Ciout           ;lo trasmette
 rts
```

FormFeed

Funzione: Trasmette alla stampante il comando form feed che fa avanzare la carta fino al foglio successivo.

Chiamata da: PrintBuffer

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, e diverse variabili utilizzate da Ciout

Sinossi: Trasmette alla stampante il comando form feed che fa avanzare la carta fino al foglio successivo.

```
FormFeed:
  lda          #FF                ;il comando form feed
  jsr          Ciout              ;lo trasmette
  rts
```

Rotate

Funzione: Ruota gli otto byte della matrice grafica puntata da r3. In questo modo il nuovo buffer da otto byte, prntBlCard, contiene la matrice grafica opportunamente riordinata per essere trasmessa alla testina della stampante.

Chiamata da: PrintPrintBuffer

Parametri: r3 indirizzo della matrice grafica in memoria

Restituisce: prntBlCard questo buffer contiene la matrice grafica ruotata

Distrugge: a, x, y

Sinossi: L'ennesimo byte memorizzato nel buffer prntBlCard è composto dai bit in posizione ennesima, prelevati da ognuno dei byte di cui è composta la matrice grafica originale puntata da r3. La matrice ruotata può quindi essere trasmessa alla stampante.

```
Rotate:
 sei                           ;disabilita gli interrupt in modo da diminuire
                               ;il tempo d'esecuzione della routine
 ldy       #7                  ;inizializza l'indice per scorrere la matrice
                               ;originale
10$:
 lda       (r3), y             ;preleva il byte dalla matrice grafica
 ldx       #7                  ;inizializza l'indice per la matrice stampabile
20$:
 ror       a                   ;memorizza in C il bit meno significativo
 ror       prntBlCard, x       ;e lo fa entrare nel byte correntemente puntato
                               ;all'interno del buffer
 dex                           ;bit successivo
 bpl       20$                 ;se non e' l'ultimo, riprendi il loop
 dey                           ;byte successivo
 bpl       10$                 ;se non e' l'ultimo, riprendi il loop
 cli                           ;riabilita gli interrupt
 rts

PRINTEND:                      ;ultima label nel driver per la stampante Epson FX-80
```

Il file geosUtilities per i driver di stampa

In questo file sono riportate le seguenti routine:

OpenFile:	apre la struttura Commodore del file
CloseFile:	chiude la struttura Commodore del file
OpenPrint:	prepara la stampante all'ascolto sul bus seriale
ClosePrint:	termina la comunicazione con la stampante sul bus seriale
Strout:	trasmette una stringa di byte sul bus seriale.

Per ottenere maggiori informazioni sul protocollo di trasmissione del bus seriale del C-64 e le sue caratteristiche, consultare la guida operativa ufficiale.

OpenFile

Parametri: Nessuno

Chiamata da: PrintBuffer

Distrugge: a, x, y

Sinossi: Questa routine prepara la struttura del file per le comunicazioni attraverso il bus seriale.

```
OpenFile:
 lda        #PRINTADDR        ;numero di dispositivo
 jsr        Listen            ;indirizza la stampante
 lda        #SECADD|$F0       ;carica l'indirizzo secondario per questa stampante
 jsr        Second            ;lo trasmette
 jsr        Unlsn             ;comanda alla stampante di interrompere l'ascolto
                              ;sul bus
 rts
```

CloseFile

Parametri: Nessuno

Chiamata da: PrintBuffer

Distrugge: a, x, y

Sinossi: Disabilita la struttura del file precedentemente aperto per comunicare con la stampante.

```
CloseFile:
 lda        #PRINTADDR      ;numero del dispositivo
 jsr        Listen          ;indirizza la stampante
 lda        #SECADD|$E0     ;carica l'indirizzo secondario per questa stampante
 jsr        Second          ;lo trasmette
 jsr        Unlsn           ;comanda alla stampante di terminare l'ascolto
                            ;sul bus seriale
 rts
```

OpenPrint

Parametri: Nessuno

Chiamata da: PrintBuffer

Distrugge: a, x, y

Sinossi: Inizializza la stampante all'ascolto sul bus seriale.

```
OpenPrint:
 lda        #PRINTADDR        ;numero del dispositivo
 jsr        Listen            ;indirizza la stampante
 lda        #SECADD|$60       ;carica l'indirizzo secondario per questa stampante
 jsr        Second            ;lo trasmette
 rts
```

ClosePrint

Parametri: Nessuno

Chiamata da: PrintBuffer

Distrugge: a, x, y

Sinossi: Comanda alla stampante di cessare l'ascolto sul bus seriale.

```
ClosePrint:
  jsr          Unlsn                  ;comanda alla stampante di cessare l'ascolto
                                      ;sul bus seriale
  rts
```

Strout

Parametrì: r3 inizio della stringa da trasmettere (i dati devono essere disposti a partire dall'ultimo)
a numero di byte da trasmettere

Chiamata da: InitPrinter

Accede a: sCount

Restituisce: sCount $FF
y $00

Distrugge: a

Sinossi: Strout (string out) trasmette la stringa di caratteri puntata da r3 e composta da a byte.

```
Strout:
 sta        sCount              ;salva l'indice della tavola
 dec        sCount
10$:
 ldy        sCount              ;riprende l'indice
 lda        ($r3),y             ;preleva il byte
 jsr        Ciout               ;lo spedisce
 dec        sCount              ;aggiorna l'indice
 bpl        10$                 ;se la tavola non e' finita, procedi
                                ;con il prossimo carattere
 rts
```

19 IL DRIVER DI STAMPA COMMODORE

Nella sua struttura generale, il driver Commodore presenta molte somiglianze con il driver Epson che abbiamo esaminato nel capitolo precedente, ma in questo caso la presenza di una testina da 7 punti rende più laborioso il processo di trasferimento dei dati grafici. Quando si hanno a disposizione stampanti a 7 punti verticali, il driver non può limitarsi a ruotare le matrici grafiche, ma deve gestire righe grafiche formate da 8 linee di scansione trasmettendo sul bus seriale solo 7 linee alla volta. Deve quindi preoccuparsi di non perdere informazioni durante la stampa. In pratica, per ogni gruppo di 7 linee stampate ne rimane una in sospeso: quando il numero delle linee in sospeso arriva a 8, il driver, prima di procedere con la riga grafica successiva, stampa l'intera riga che si è venuta a creare durante il processo di trasferimento. A questo punto però è rimasta ancora una linea da stampare. Quindi il ciclo non riprende più dalla fase iniziale, nella quale non vi è nessuna linea in avanzo, ma dalla fase appena successiva. In pratica, quando si inizia la stampa di una pagina, il driver affronta una fase preliminare esterna al ciclo, che nel resto della pagina non verrà più ripercorsa. Il procedimento ciclico viene eseguito dalle routine TopRollBuffer e BotRollBuffer. Per la precisione, TopRollBuffer chiama RollaCard per prelevare il primo byte (quello più in alto) di una matrice grafica memorizzata nel buffer utente, spostando verso l'alto di una posizione i byte rimanenti.

Ogni trasferimento di una riga grafica al bus seriale, provoca l'incremento di una unità tra le righe "in sospeso" conservate nel buffer utente. Nel driver il contatore delle linee in sospeso è breakCount. Queste linee vengono inserite nella riga grafica successiva, e trasmesse non appena PrintBuffer viene nuovamente chiamata.

Quando sono state stampate 7 linee di scansione, BotRollBuffer trasferisce le linee di scansione avanzate dal buffer utente nel buffer di stampa interno facendo opportunamente salire la linea avanzata dall'ultima stampa, che deve poi diventare la prima della nuova riga grafica. Ovviamente queste linee, essendo le prime che devono essere stampate con la successiva chiamata di PrintBuffer, verranno disposte in alto nel buffer di stampa. Per meglio comprendere il procedimento di stampa, analizziamo un esempio pratico osservando le tavole riportate nelle pagine che seguono. In esse sono rappresentati sei stadi, i primi e gli ultimi all'interno di un ciclo di nove stadi. I tre stadi centrali sono facilmente deducibili una volta che si è compreso il funzionamento dell'algoritmo di stampa. L'esempio descritto nelle tavole, per semplicità, assume che la pagina da stampare contenga soltanto una fila verticale di 9 matrici grafiche.

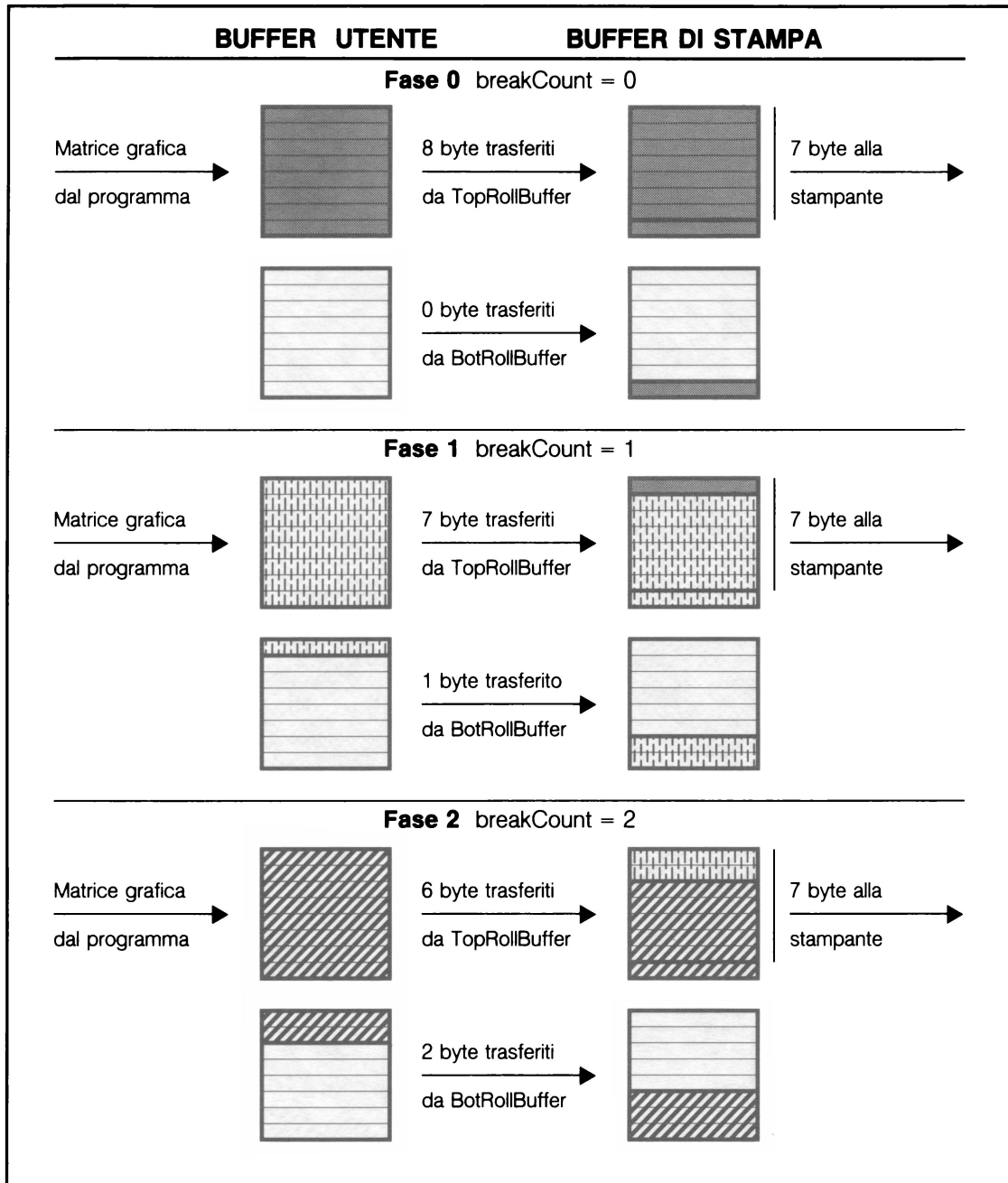
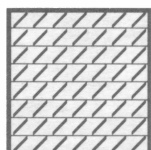
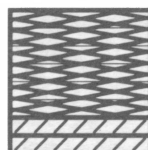
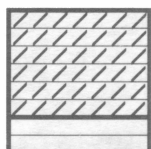
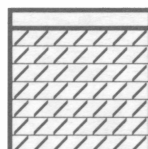
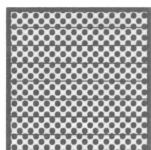
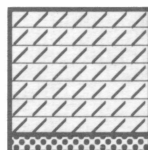
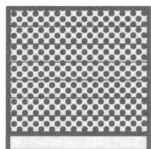
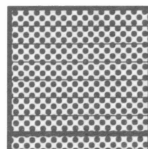
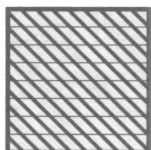
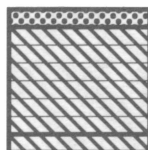
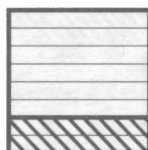
Quando l'applicazione inizia la stampa della pagina, entra nella fase 0, caratterizzata dall'assenza di linee in avanzo e quindi da breakCount = 0. L'applicazione memorizza nel buffer utente la prima matrice (che nel nostro esempio rappresenta l'intera riga grafica). Tramite TopRollBuffer, il driver trasferisce 8 (8 = 8 – breakCount) nel buffer di stampa, cioè l'intera matrice grafica. Subito dopo, 7 degli 8 byte vengono stampati, lasciando nel buffer di stampa un byte di avanzo. Dal momento che in questa fase preliminare non vi erano byte in avanzo nel buffer utente (breakCount = 0), BotRollBuffer non esegue alcuna operazione.

L'applicazione riprende il controllo e memorizza nel buffer utente una nuova matrice che sul foglio dev'essere stampata appena sotto la precedente. breakCount ora vale 1, e si entra nella prima fase del ciclo. Tramite TopRollBuffer, il driver trasferisce 7 byte (8 – breakCount) dal buffer utente al buffer di stampa, facendo salite il byte in avanzo presente nel buffer di stampa al primo posto. In questo modo rimane un byte non trasferito nel buffer utente. Vengono stampati i primi 7 byte e nel buffer di stampa rimane, come sempre, un byte in avanzo. Vi sono ora due byte in avanzo: uno nel buffer utente e uno nel buffer di stampa. A questo punto BotRollBuffer trasferisce un byte (1 = breakCount) dal buffer utente al buffer di stampa. Questo trasferimento fa in modo che il byte in avanzo presente nel buffer di stampa salga di una posizione.

L'applicazione riceve nuovamente il controllo e memorizza una nuova matrice grafica nel buffer utente, entrando così nella seconda fase del ciclo caratterizzata da breakCount = 2.

Il ciclo continua fino al termine della settima fase, quando i byte totali sono 8, e sono tutti contenuti nel buffer di stampa. Allora il driver stampa anche questo gruppo di 8 byte, producendo nuovamente 1 byte in avanzo. breakCount viene aggiornato nuovamente con il valore 1 e il ciclo riprende dalla prima fase, e non più dalla fase 0.

Questo è l'algoritmo di stampa impiegato dal driver per le stampanti Commodore compatibili a 7 punti. Com'è possibile rendersi conto, dal punto di vista dell'applicazione il tipo di stampante impiegata, e quindi di driver, è completamente irrilevante.

Procedura di stampa a 7 bit

(Le fasi dalla 3 alla 5 non sono riportate per brevità)
BUFFER UTENTE
BUFFER DI STAMPA
Fase 6 breakCount = 6
Matrice grafica
dal programma
2 byte trasferiti
da TopRollBuffer
7 byte alla
stampante
6 byte trasferiti
da BotRollBuffer
Fase 7 breakCount = 7
Matrice grafica
dal programma
1 byte trasferito
da TopRollBuffer
7 byte alla
stampante
7 byte trasferiti
da BotRollBuffer
7 byte alla
stampante
Fase 1 breakCount = 1
Matrice grafica
dal programma
7 byte trasferiti
da TopRollBuffer
7 byte alla
stampante
1 byte trasferito
da BotRollBuffer

Il driver di stampa per le stampanti Commodore compatibili

Questo driver è stato ideato per le stampanti:

COMMODORE MPS-801, MPS-803, MPS-1000, 1525
ERGO SYSTEMS HUSH 80CD
OKIDATA OKIMATE 10
SEIKOSHA SP-1000VC

ed è stato sottoposto al test di verifica utilizzando le stampanti:

COMMODORE MPS-801, MPS-1000
ERGO SYSTEMS HUSH 80CD
SEIKOSHA SP-1000VC.

Ecco il listato del file sorgente utilizzato per questo driver di stampa.

Costanti per la stampante

```
LOWERCASE       = 7             ;comando per la stampante
CARDSWIDE       = 60            ;larghezza in matrici grafiche Commodore
CARDSDEEP       = 94            ;altezza in matrici grafiche Commodore
SECADD          = LOWERCASE     ;indirizzo secondario
CGPX            = 8             ;comando di attivazione del modo grafico
ECGPX           = 15            ;comando di disattivazione del modo grafico
PRINTADDR       = 4             ;numero di dispositivo della stampante
PRINTBASE       = $7900         ;indirizzo al quale il driver dev'essere rilocato

                                ;per questi file consultare l'appendice
.include        geosMacros      ;macro per l'Assembler della Berkeley
.include        geosConstants   ;costanti
.include        geosMemoryMap   ;locazioni e variabili nella RAM
.include        geosRoutines    ;jump table per le routine di GEOS

.psect          PRINTBASE       ;indirizzo al quale allocare il file oggetto compilato
```

Jump table residente

```
InitForPrint:     rts                           ;prima routine a PRINTBASE
                  nop                           ;per questa stampante la routine
                  nop                           ;non deve compiere alcuna operazione

StartPrint:       jmp   r_StartPrint            ;seconda routine a PRINTBASE + 3
PrintBuffer:      jmp   r_PrintBuffer           ;terza routine a PRINTBASE + 6
StopPrint:        jmp   r_StopPrint             ;quarta routine a PRINTBASE + 9
GetDimensions:    jmp   r_GetDimensions         ;quinta routine a PRINTBASE + 12
PrintASCII:       jmp   r_PrintASCII            ;sesta routine a PRINTBASE + 15
StartASCII:       jmp   r_StartASCII            ;settima routine a PRINTBASE + 18
SetNLQ:           rts                           ;ottava routine a PRINTBASE + 21
                  nop                           ;non disponibile per questo tipo di stampante
                  nop
```

Variabili in RAM e utility

```
printerName:      .byte "Comm. Compat.",0       ;nome della stampante come dovrebbe apparire
                                                ;nel menu di selezione della stampante
prntBlCard:       .byte 0,0,0,0,0,0,0,0         ;matrice grafica stampabile
breakCount:       .byte 0                       ;contatore delle linee rimaste
```

```
sCount:          .byte 0        ;contatore delle linee all'interno della matrice
                                ;grafica corrente
cardCount:       .byte 0        ;contatore delle matrici grafiche azzerate a fine riga
modeFlag:        .byte 0        ;$00=stampa grafica, $FF=stampa ASCII

.include         geosUtilities  ;file contenente le routine di basso livello
                                ;di comunicazione con la stampante. Questo file e'
                                ;riportato alla fine del capitolo precedente
```

Le routine direttamente accessibili dall'applicazione

StartPrint

Funzione: Prepara la stampante all'inizio di ogni pagina del documento.

Parametri: Nessuno

Restituisce: x ≠ 0 codice dell'eventuale errore

Distrugge: Tutti i registri

Sinossi: Questa routine prepara il bus seriale e la stampante per procedere alla stampa di una pagina. La stampante viene opportunamente configurata e il contatore breakCount viene azzerato.

```
r_StartPrint:
   lda      #0              ;imposta il modo grafico
   sta      modeFlag
StartIn:
   lda      #PRINTADDR      ;attiva il dispositivo 4
   jsr      SetDevice
   jsr      InitForIO       ;attiva il Kernel del C-64, lo spazio di I/O
                            ;e disabilita gli interrupt
   lda      #0
   sta      breakCount      ;inizializza il contatore delle linee rimaste
   sta      $90             ;imposta l'indicatore d'errore a nessun errore
   jsr      OpenFile        ;apre il file fittizio di comunicazione con
   lda      $90             ;la stampante se si e' verificato un errore con
                            ;il canale di output, esegue la routine
   bne      20$             ;di gestione degli errori
```

```
    jsr         Delay           ;attende per eliminare eventuali problemi
                                ;di temporizzazione
    jsr         DoneWithIO      ;ripristina lo stato precedente alla chiamata
                                ;di InitForIO
    ldx         #0              ;nessun errore
    rts
20$:
    pha                         ;salva temporaneamente l'eventuale codice d'errore
                                ;bit 0 impostato a 1: timeout, scrittura
                                ;bit 7 impostato a 1: device non presente
    jsr         CloseFile       ;chiude il file
    jsr         DoneWithIO      ;ripristina lo stato precedente alla chiamata
                                ;di InitForIO
    pla                         ;recupera il codice d'errore
    tax                         ;e lo trasferisce in x
    rts
Delay:
    ldx         #0
10$:
    ldy         #0
20$:
    dey
    bne         20$
    dex
    bne         10$
    rts
```

PrintBuffer

Funzione: Trasmette alla stampante i dati grafici contenuti nel buffer da 640 byte (80 matrici grafiche) creato dall'applicazione.

Parametri:
r0 indirizzo del buffer da 640 byte (80 matrici grafiche)
r1 indirizzo del buffer di stampa da 1920 byte che l'applicazione deve mettere a disposizione della routine PrintBuffer

Nota: l'indirizzo di quest'ultimo buffer non deve cambiare fra una chiamata a PrintBuffer e la successiva. I driver di stampa per le stampanti a 7 bit utilizzano questo buffer per non perdere le linee di scansione accumulate a ogni chiamata di PrintBuffer

Restituisce: r0, r1 inalterati

Distrugge: Tutti i registri

Sinossi: PrintBuffer, come è già stato spiegato, è la routine di livello più alto in grado di trasferire il contenuto grafico del buffer da 640 byte dalla memoria del C-64 alla stampante attraverso la porta seriale.

```
r_PrintBuffer:
   lda        #PRINTADDR
   jsr        SetDevice          ;attiva il dispositivo associato
   jsr        InitForIO          ;attiva il Kernel del C-64), lo spazio di I/O
                                 ;e disabilita gli interrupt
   jsr        OpenPrint          ;apre il canale di comunicazione con la stampante
   jsr        I_PrintBuffer      ;trasmette la riga grafica alta 8 bit memorizzata
                                 ;nel buffer utente
   jsr        ClosePrint         ;chiude il canale di comunicazione con la stampante
   jsr        DoneWithIO         ;ripristina lo stato precedente alla chiamata
                                 ;di InitForIO
   rts
```

```
I_PrintBuffer:
   jsr       TopRollBuffer         ;trasferisce (8 - breakCount) linee dal buffer
                                   ;utente al buffer di stampa
   MoveW     r1, r3
   jsr       PrintPrintBuffer      ;stampa il contenuto del buffer interno
   jsr       BotRollBuffer         ;trasferisce breakCount linee non stampate
                                   ;dal buffer utente al buffer di stampa
   lda       breakCount            ;controlla se le linee rimaste sono 8
   cmp       #7
   bne       5$                    ;se non sono 8, salta
   MoveW     r1, r3                ;punta al buffer di stampa
   jsr       PrintPrintBuffer      ;stampa la riga grafica che si e' creata con
                                   ;le successive linee "in sospeso" e rigenera
                                   ;una linea in sospeso
   lda       #0                    ;azzera il contatore delle linee rimaste
   sta       breakCount
5$:
   inc       breakCount            ;incrementa il contatore di linee in sospeso,
                                   ;in modo che la fase successiva non sia
                                   ;la condizione iniziale (breakCount = 0), ma
                                   ;quella seguente (breakCount = 1)
   rts
```

StopPrint

Funzione: Si esegue al termine della stampa di ogni pagina, per cancellare il buffer di stampa e ordinare alla stampante di fare avanzare la carta fino al foglio successivo. StopPrint viene eseguita anche quando termina il documento.

Parametri: r0 indirizzo del buffer da 640 byte (80 matrici grafiche)
r1 indirizzo del buffer di stampa da 1920 byte che l'applicazione deve mettere a disposizione del driver per suo uso interno

Nota: l'indirizzo di quest'ultimo buffer non deve cambiare fra una chiamata a PrintBuffer e la successiva. I driver di stampa per le stampanti a 7 bit utilizzano questo buffer per conservarvi le linee di scansione perse a ogni chiamata di PrintBuffer

Restituisce: r0, r1 inalterati

Distrugge: Tutti i registri

Sinossi: StopPrint dev'essere chiamata quando si giunge al termine di una pagina o dell'intero documento. Manda in esecuzione SetDevice, InitForIO, mette in ascolto la stampante, e se la testina è a 7 bit stampa eventuali linee rimaste nel buffer di stampa. Quindi cancella il buffer di stampa, effettua l'avanzamento del foglio all'inizio della nuova pagina, comanda alla stampante di cessare l'ascolto sul bus seriale, chiude il file Commodore di output ed esegue la routine DoneWithIO.

```
r_StopPrint:
    lda     #PRINTADDR
    jsr     SetDevice       ;attiva il dispositivo associato
    jsr     InitForIO       ;attiva il Kernel del C-64, lo spazio di I/O
                            ;e disabilita gli interrupt
    jsr     OpenPrint       ;apre il canale di comunicazione con la stampante
    bit     modeFlag        ;se la stampa e' ASCII
    bmi     10$             ;effettua lo svuotamento del buffer
    PushW   r0              ;salva gli indirizzi dei buffer
```

```
	PushW	r1
	MoveW	r0, r1		;carica l'indirizzo del buffer da cancellare
	LoadW	r0,#640		;lunghezza in byte del buffer da cancellare
	jsr	ClearRam	;lo cancella
	PopW	r1		;recupera gli indirizzi dei buffer
	PopW	r0
	jsr	I_PrintBuffer	;stampa le linee accumulate nel buffer
10$:
	jsr	FormFeed	;fa avanzare la carta fino all'inizio
				;del nuovo foglio
	jsr	ClosePrint	;chiude il canale di stampa
	jsr	CloseFile	;chiude il file di output
	jsr	DoneWithIO	;ripristina lo stato precedente alla chiamata di
				;InitForIO
	rts
```

GetDimensions

Funzione: Restituisce le dimensioni (in matrici grafiche) del disegno che la stampante è in grado di riprodurre sul foglio.

Parametri: Nessuno

Restituisce:
- x larghezza in matrici grafiche della linea che la stampante è in grado di stampare
- y altezza in matrici grafiche dello spazio lungo il foglio entro il quale la stampante è in grado di stampare

Distrugge: Niente

Sinossi: GetDimensions restituisce il numero di matrici grafiche in larghezza e in altezza che la stampante è in grado di stampare in un rettangolo di 8 x 10,5 pollici. Questo rettangolo è compreso interamente nel foglio di stampa, il quale misura di solito 8,5 x 11 pollici.

```
r_GetDimensions:
   ldx       #CARDSWIDE        ;numero di matrici grafiche orizzontali
   ldy       #CARDSDEEP        ;numero di matrici grafiche verticali
   lda       #0                ;impostato a 0 per i driver grafici
   rts
```

StartASCII

Funzione: Inizializza la stampante Commodore in modo che tratti i dati trasmessi come codici ASCII.

Parametri: Nessuno

Restituisce: Niente

Distrugge: Tutti i registri

Sinossi: Imposta il flag modeFlag per indicare la selezione del modo ASCII di stampa. Chiama StartIn per procedere all'inizializzazione. Questa routine dev'essere chiamata all'inizio di ogni pagina.

```
r_StartASCII:
 LoadB       modeFlag, #$FF       ;imposta il modo ASCII di stampa
 jmp         StartIn              ;utilizza una parte di StartPrint per procedere
                                  ;alla preparazione della stampante
```

PrintASCII

Funzione: Trasmette alla stampante una stringa ASCII a terminazione nulla.

Parametri: r0 puntatore alla stringa ASCII

Restituisce: Niente

Distrugge: Tutti i registri

Sinossi: Trasmette la stringa ASCII a terminazione nulla puntata da r0. La stringa può contenere caratteri CR per la separazione delle righe. La routine, quando incontra il carattere CR, trasmette di seguito anche il carattere LF.

```
r_PrintASCII:
   lda       #PRINTADDR
   jsr       SetDevice          ;attiva il dispositivo associato
   jsr       InitForIO          ;attiva il Kernel del C-64, lo spazio di I/O
                                ;e disabilita gli interrupt
   jsr       OpenPrint          ;apre il canale di comunicazione con la stampante
10$:
   ldy       #0                 ;azzera l'indice per scorrere la stringa ASCII
   lda       (r0), y            ;preleva il carattere
   beq       30$                ;se ha raggiunto la fine della stringa, conclude
                                ;la sua esecuzione
   cmp       #65                ;controlla se il carattere e' alfanumerico
   bcc       12$                ;salta la conversione se e' minore di 'A'
   cmp       #123
   bcs       12$                ;salta la conversione se e' maggiore di 'z'
   eor       #$20               ;converte le maiuscole in minuscole e viceversa
12$:
   jsr       Ciout
```

```
        IncW        r0                  ;punta al carattere successivo
        jmp         10$                 ;ripete il loop
30$:
        jsr         ClosePrint          ;chiude il canale con la stampante
        jsr         DoneWithIO          ;ripristina lo stato precedente alla chiamata
                                        ;di InitForIO
        rts
```

Le routine interne del driver

PrintPrintBuffer

Funzione: Trasmette il buffer di stampa puntato da r3.

Chiamato da: PrintBuffer

Parametri: r3 indirizzo del buffer in memoria

Restituisce: Niente

Distrugge: a, x, y, r0 - r15

Sinossi: Tramite TestBuffer controlla se il buffer è vuoto, prima di procedere alla stampa della riga grafica. Se il buffer non è vuoto, la routine, per ogni matrice grafica prelevata dal buffer, ruota i byte e li trasmette alla stampante. TestBuffer inoltre restituisce in cardCount il numero di matrici grafiche azzerate a partire dal lato destro della riga grafica. PrintPrintBuffer, valutando opportunamente cardCount, è in grado di ottimizzare il tempo di stampa trasmettendo esclusivamente le matrici grafiche significative della riga.

```
PintPrintBuffer:
   PushW       r3                   ;salva il puntatore al buffer
   jsr         TestBuffer           ;controlla se il buffer e' completamente azzerato
                                    ;e aggiorna cardCount
   bcs         5$                   ;se il buffer contiene dati, inizia
                                    ;il trasferimento
   PopW        r3                   ;svuota lo stack
   jsr         SetGraphics          ;attiva il modo grafico
   jmp         20$                  ;termina la stampa della riga
5$:
   jsr         SetGraphics          ;attiva il modo grafico per la riga corrente
```

```
    PopW        r3              ;recupera il puntatore alla matrice grafica
                                ;corrente
    lda         #CARDSWIDE      ;carica la massima larghezza della riga grafica

    sec
    sbc         cardCount       ;sottrae il numero di mappe grafiche nulle
                                ;a partire dal lato destro della riga grafica
    tax                         ;ottenendo l'esatto numero di matrici grafiche
                                ;significative
10$:
    txa                         ;salva x
    pha
    jsr         Rotate          ;ruota la matrice grafica puntata da r3
    jsr         SendBuff        ;trasmette la matrice grafica ruotata
    AddVW       #8, r3          ;aggiorna r3 in modo che punti alla matrice
                                ;grafica successiva
    pla                         ;recupera x
    tax
    dex
    bne         10$             ;se non ha trasmesso tutte le matrici grafiche,
                                ;ripete l'operazione con la successiva
20$:
    jsr         Greturn         ;effettua il ritorno del carrello a nuova riga
    jsr         UnSetGraphics   ;disabilita il modo grafico
    rts
```

TopRollBuffer

Funzione: Trasferisce dal buffer utente al buffer di stampa il numero di linee indicato da (8 – breakCount).

Chiamata da: PrintBuffer

Parametri: r0 puntatore al buffer utente
r1 puntatore al buffer di stampa

Restituisce: Niente

Distrugge: a, x

Sinossi: TopRollBuffer trasferisce tante linee quante ne indica (8 – breakCount) dal buffer utente al buffer di stampa. Sono le linee appena memorizzate dall'applicazione. Se è presente un residuo di linee prodotto dalla stampa delle righe precedenti, la routine provvede a posizionare nel buffer di stampa le nuove linee in modo che non cancellino quelle rimaste, già trasferite da BotRollBuffer.

```
TopRollBuffer:
   PushW       r0                   ;salva i puntatori ai buffer
   PushW       r1
   ldx         #CARDSWIDE-1         ;carica il numero di matrici grafiche diminuito
                                    ;di 1
10$:
   ldy         breakCount           ;preleva il contatore delle linee rimaste
   lda         topBreakTab, y       ;preleva il numero di linee da trasferire
   jsr         RollaCard            ;trasferisce la matrice grafica
   dex                              ;se le matrici grafiche non sono ancora finite
   bpl         10$                  ;riprende il loop per trasferire la matrice
                                    ;successiva
   PopW        r1                   ;ripristina i puntatori
   PopW        r0
   rts
topBreakTab:
   .byte       8, 7, 6, 5, 4, 3, 2, 1
```

BotRollBuffer

Funzione: Trasferisce nel buffer di stampa le breakCount linee rimaste.

Chiamata da: PrintBuffer

Parametri: r0 puntatore al buffer utente
r1 puntatore al buffer di stampa

Restituisce: Niente

Distrugge: a, x

Sinossi: BotRollBuffer trasferisce le linee rimaste nel buffer di stampa, collocandole nei primi posti, in modo che TopRollBuffer possa poi completare il buffer di stampa trasferendovi le linee della riga trasmessa dall'applicazione.

```
BotRollBuffer:
   lda       breakCount        ;se breakCount = 0, la routine non deve eseguire
   beq       20$               ;alcuna operazione
                               ;alcuni driver per stampanti a 7 punti
                               ;non contengono questo controllo, e producono
                               ;un difetto nella prima riga di stampa
   PushW     r0                ;salva i puntatori ai buffer
   PushW     r1
   ldx       #CARDSWIDE-1      ;carica il numero delle matrici grafiche diminuito
                               ;di 1
10$:
   lda       breakCount        ;carica il contatore delle linee in sospeso
                               ;non ancora trasferite
   jsr       RollaCard         ;trasferisce il numero di linee della matrice
                               ;grafica indicato nel registro a
   dex
   bpl       10$               ;se le matrici grafiche non sono finite,
                               ;riprendi il loop
   PopW      r1                ;ripristina i puntatori
   PopW      r0
20$:
   rts
```

RollaCard

Funzione: Trasferisce dal buffer utente al buffer di stampa tanti byte quanti ne sono indicati nel registro a. Ogni byte appartiene a una diversa linea di stampa.

Chiamata da: TopRollBuffer, BotRollBuffer

Parametri: a numero di linee da trasferire

Restituisce: r0 = r0 + 8
r1 = r1 + 8
r3L = 0

Distrugge: a

Sinossi: Trasferisce dal buffer utente al buffer di stampa tanti byte quanti ne sono indicati nel registro a. Ogni byte appartiene a una diversa linea di stampa.

```
RollaCard:
   sta        r3L                  ;memorizza il numero di loop da eseguire
                                   ;(cioe' il numero di linee da trasferire)
10$:
   jsr        Roll8bytesOut        ;trasferisce nel buffer di stampa un byte
                                   ;della matrice grafica memorizzata
   jsr        Roll8bytesIn         ;nel buffer utente
   dec        r3L                  ;se non sono stati trasferiti tutti i byte
                                   ;della matrice grafica corrente
   bne        10$                  ;riprendi il loop
   AddVW      #8, r0               ;aggiorna il puntatore del buffer utente
                                   ;alla nuova matrice grafica
   AddVW      #8, r1               ;aggiorna il puntatore del buffer di stampa
                                   ;alla nuova matrice grafica
   rts
```

TestBuffer

Funzione: Controlla se il buffer contiene dati grafici da stampare. La routine restituisce in cardCount il numero di matrici grafiche azzerate che incontra lungo la riga grafica scorrendola da destra a sinistra. Questa informazione serve a evitare che PrintPrintBuffer perda tempo nella trasmissione di matrici grafiche non significative, che servono solo per completare la riga.

Chiamata da: PrintPrintBuffer

Parametri: r3 puntatore all'inizio del buffer da controllare

Restituisce: c impostato a 1 se il buffer contiene dati, impostato a 0 se il buffer è vuoto

Distrugge: a, x, r3

Sinossi: Controlla che i byte contenuti nel buffer non siano tutti a 0. Il controllo inizia dal lato destro della riga grafica e procede verso sinistra, matrice per matrice. Durante la scansione, la routine mantiene aggiornato il contatore countCard. Quando incontra un byte diverso da 0, la routine cessa la scansione, imposta a 1 il carry ed esce. Se invece la routine non incontra alcun byte diverso da zero, esce impostando il carry a 0. Per questo driver, TestBuffer controlla solo matrici da 7 byte.

```
TestBuffer:
 LoadB      cardCount, #0            ;azzera il contatore delle matrici grafiche azzerate
 lda        #6                       ;assume che la testina sia da 7 punti
 sta        sCount                   ;aggiorna il contatore di linee di scansione
                                     ;all'interno della matrice grafica
 AddVW      #((CARDSWIDE-1)*8), r3   ;aggiorna r3 in modo che punti al primo byte
                                     ;dell'ultima matrice grafica lungo la riga
 ldx        #CARDSWIDE               ;aggiorna x con il numero di matrici presenti
                                     ;in una riga grafica
10$:
 ldy        sCount                   ;preleva il contatore di linee all'interno
                                     ;della matrice grafica corrente
```

```
20$:
   lda      (r3), y        ;controlla un byte
   bne      30$            ;se e' uguale a 0 termina l'esecuzione della routine
   dey                     ;punta al byte precedente nella matrice grafica
   bpl      20$            ;se non ha terminato la scansione della matrice
                           ;grafica corrente, controlla il byte precedente
   SubVW    #8, r3         ;aggiorna r3 in modo che punti all'inizio
                           ;della matrice grafica precedente
   inc      cardCount      ;incrementa il contatore di matrici grafiche
                           ;azzerate
   dex                     ;se non sono state controllate tutte le matrici
                           ;grafiche della riga
   bpl      10$            ;va a verificare la precedente
   clc                     ;se con questa istruzione la routine conclude la sua
                           ;esecuzione, significa che il buffer e' vuoto
   rts
30$:
   sec                     ;imposta il carry a 1 per indicare che il buffer
                           ;non e' vuoto
   rts
```

Roll8bytesIn

Funzione: Trasla di una posizione verso l'alto i byte della matrice grafica corrente memorizzata nel buffer di stampa e memorizza nell'ultima posizione il valore passato in a.

Chiamata da: RollaCard

Parametri:
a byte da aggiungere nell'ultima posizione all'interno della matrice grafica
r1 puntatore alla matrice grafica da traslare verso l'alto di una posizione

Restituisce: Niente

Distrugge: a, y

Sinossi: Roll8bytesIn trasla di una posizione verso l'alto i byte della matrice grafica memorizzata nel buffer di stampa. L'ultima posizione che si crea viene riempita con il byte passato alla routine nel registro a. Tale byte dev'essere preparato da Roll8bytesOut che di solito viene chiamata immediatamente prima di Roll8bytesIn.

```
Roll8bytesIn:
   pha                             ;salva il byte da aggiungere
   ldy       #0                    ;inizializza l'indice per scorrere i byte
                                   ;della matrice
10$:
   iny                             ;punta alla successiva posizione verso il basso
                                   ;(il primo byte viene perso)
   lda       (r1), y               ;carica dalla matrice il byte della linea
   dey                             ;punta alla posizione precedente
   sta       (r1), y               ;e vi memorizza il byte
   iny                             ;punta alla successiva linea verso il basso
   cpy       #7                    ;se non siamo all'ultimo byte della matrice
   bmi       10$                   ;riprende il loop
   pla                             ;riprende il byte da aggiungere
   sta       (r1), y               ;e lo memorizza nell'ultima posizione
                                   ;della matrice grafica
   rts
```

Roll8bytesOut

Funzione: Restituisce in a il primo byte della matrice grafica corrente memorizzata nel buffer utente e trasla di una posizione verso l'alto i restanti byte.

Chiamata da: RollaCard

Parametri: r0 puntatore alla matrice grafica da traslare verso l'alto di una posizione

Restituisce: a primo byte della matrice grafica corrente

Distrugge: a, y

Sinossi: Roll8bytesOut trasla di una posizione verso l'alto i byte della matrice grafica corrente memorizzata nel buffer utente. Viene utilizzata per svuotare il buffer utente. Con la traslazione viene restituito il primo byte che in seguito viene passato a Roll8bytesIn.

```
Roll8bytesOut:
   ldy        #0             ;inizializza l'indice per scorrere i byte
                             ;della matrice
   lda        (r0), y        ;carica dalla matrice il byte piu' alto
   pha                       ;lo salva
10$:
   iny                       ;punta alla successiva posizione verso il basso
   lda        (r0), y        ;carica il byte della linea dalla matrice
   dey                       ;punta alla posizione precedente
   sta        (r0), y        ;e vi memorizza il byte
   iny                       ;punta alla successiva linea verso il basso
   cpy        #7             ;se non siamo all'ultimo byte della matrice
   bmi        10$            ;riprende il loop
   pla                       ;riprende il byte da passare a Roll8bytesIn
   rts
```

Le routine specifiche Commodore

SetGraphics
UnSetGraphics

Funzione: SetGraphics attiva il modo grafico della stampante.
UnSetGraphics disattiva il modo grafico.

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, le variabili utilizzate da Ciout

```
SetGraphics:
   lda        #CGPX              ;comando per abilitare il modo grafico
   jsr        Ciout              ;trasmette il comando
   rts
UnSetGraphics:
   lda        #ECGPX             ;comando per disabilitare il modo grafico
   jsr        Ciout              ;trasmette il comando
   rts
```

SendBuff

Funzione: Trasmette una matrice grafica stampabile attraverso la porta seriale.

Chiamata da: PrintPrintBuffer

Parametri: prntBlCard questo buffer deve contenere la matrice grafica già ruotata

Restituisce: Niente

Distrugge: a, x

Sinossi: Quando una matrice grafica è stata ruotata in modo che i bit di ogni byte corrispondano ai punti verticali della testina di stampa, SendBuff invia i dati della matrice alla stampante attraverso il bus seriale.

```
SendBuff:
 ldx        #0                  ;inizializza il contatore
10$:
 txa                            ;salva il contatore
 pha
 lda        prntBlCard, x       ;preleva il byte da trasmettere
 ora        #$80                ;imposta a 1 il bit 7 che non viene stampato
 jsr        Ciout               ;trasmette il byte
 pla                            ;recupera il contatore
 tax
 inx                            ;punta al nuovo byte
 cpx        #8                  ;se non sono stati trasmessi tutti i byte
                                ;della matrice
 bne        10$                 ;riprende il loop di trasmissione
 rts
```

Greturn

Funzione: Trasmette alla stampante il carattere CR (ritorno carrello).

Chiamata da: PrintBuffer

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, e diverse variabili utilizzate da Ciout

Sinossi: Greturn trasmette alla stampante il carattere CR (ritorno carrello).

```
Greturn:
 lda        #CR                 ;comando di ritorno carrello
 jsr        Ciout               ;lo trasmette
 rts
```

FormFeed

Funzione: Trasmette alla stampante il comando form feed che fa avanzare la carta fino al foglio successivo.

Chiamata da: PrintBuffer

Parametri: Nessuno

Restituisce: Niente

Distrugge: a, e diverse variabili utilizzate da Ciout

Sinossi: Trasmette alla stampante il comando form feed che fa avanzare la carta fino al foglio successivo.

```
FormFeed:
  lda          #FF                ;il comando form feed
  jsr          Ciout              ;lo trasmette
  rts
```

Rotate

Funzione: Ruota gli otto byte della matrice grafica puntata da r3. In questo modo il nuovo buffer da otto byte, prntBlCard, contiene la matrice grafica opportunamente riordinata per essere trasmessa alla stampante.

Chiamata da: PrintPrintBuffer

Parametri: r3 indirizzo della matrice grafica in memoria

Restituisce: prntBlCard questo buffer contiene la matrice grafica ruotata

Distrugge: a, x, y

Sinossi: L'ennesimo byte memorizzato nel buffer prntBlCard è composto da tutti gli ennesimi bit dei byte contenuti nella matrice grafica originale puntata da r3. La matrice ruotata può quindi essere trasmessa alla stampante.

```
Rotate:
  sei                               ;disabilita gli interrupt, diminuendo cosi'
                                    ;il tempo di esecuzione della routine
  ldy        #7                     ;inizializza l'indice per scorrere la matrice
                                    ;originale
10$:
  lda        (r3), y                ;preleva il byte dalla matrice grafica
  ldx        #7                     ;inizializza l'indice per la matrice stampabile
20$:
  ror        a                      ;memorizza in C il bit meno significativo
  ror        prntBlCard, x          ;e lo fa entrare nel byte correntemente puntato
                                    ;all'interno del buffer
  dex                               ;bit successivo
  bpl        20$                    ;se non e' l'ultimo riprende il loop
  dey                               ;byte successivo
  bpl        10$                    ;se non e' l'ultimo riprende il loop
  cli                               ;riabilita gli interrupt
  rts

PRINTEND:                           ;ultima label nel driver per le stampanti
                                    ;compatibili Commodore
```

20 LA CONFIGURAZIONE DI SISTEMA

La procedura di Warm Start

Quando si chiama la routine FirstInit, viene mandata in esecuzione la cosiddetta "Warm Start" (partenza a caldo). È una procedura che riconfigura il sistema a uno stato di default, e dal momento che le applicazioni (per loro natura) devono operare in un ambiente perfettamente noto, è indispensabile che venga applicata. Qualsiasi applicazione produce inevitabilmente alcune alterazioni nella configurazione di default, che a priori non sono prevedibili. Per questo, la "Warm Start" deve di nuovo inizializzare tutte le variabili e i buffer di sistema cosicché l'ambiente operativo per la nuova applicazione sia perfettamente determinato. Il nome "Warm Start" indica che si tratta di una procedura eseguibile solo a sistema già installato e attivato. È comunque parte della routine "ColdStart" che GEOS esegue al momento dell'installazione.

Il listato che segue riporta le informazioni fondamentali sulla configurazione di default. È una raccolta delle operazioni più importanti che vengono eseguite durante la procedura di "Warm Start".

I compiti della routine FirstInit non devono essere confusi con quelli della routine StartAppl. FirstInit infatti riporta il sistema alla condizione in cui si trova quando è appena stato installato: tutte le variabili riaggiornate, i buffer e i vettori gestiti dal Kernel. Se per esempio lo schermo ha dei colori diversi da quelli di default, il mouse ha parametri di spostamento alterati e sono previsti due drive, quando si esegue la routine FirstInit tutte queste modifiche vengono eliminate. StartAppl, invece, non ripristina *tutte* le variabili che costituiscono la configurazione corrente del sistema, ma riporta allo stato di default soltanto quelle strettamente necessarie perché l'applicazio-

ne possa iniziare la propria esecuzione in un ambiente "pulito". Oltre a questa operazione di parziale pulizia, StartAppl prepara i dati da trasferire all'applicazione, e la manda in esecuzione. Le variabile aggiornate anche da StartAppl sono evidenziate, nella tavola di "Warm Start" che segue, con le lettere "(SA)" all'inizio del commento.

Numero di byte	Valore	Indirizzo	Commento
	cld		;(SA) cancella il modo decimale della CPU
byte	#$30	CPU_DATA	;(SA) seleziona tutta la RAM disponibile ;per il C-64
$0A00	0	OS_VARS	;azzera l'area RAM per le variabili ;GEOS globali e locali

```
;Configura le variabili hardware del C-64. Imposta il banco di memoria visto
;dal processore VIC e tutte le variabili del processore 6526 CIA con i valori
;appropriati per l'ambiente GEOS
```

Numero di byte	Valore	Indirizzo	Commento
byte	#$2F	CPU_DDR	;(SA) aggiorna il registro direzione dati ;del 6510
byte	#$36	CPU_DATA	;imposta provvisoriamente la mappa ;della memoria ;in modo che siano attivati ;lo spazio di I/O e il Kernel del C-64

```
;Inizializza i valori di scansione della tastiera a nessun tasto premuto
```

Numero di byte	Valore	Indirizzo	Commento
byte	0	cia1base + $03	;(SA) azzera il registro b di direzione ;dati del CIA 1
byte	0	cia1base + $0F	;(SA) azzera il registro b di controllo ;del CIA 1
byte	0	cia2base + $0F	;(SA) azzera il registro b di controllo ;del CIA 2

```
;se NTSC utilizza $00, se PAL utilizza $80
```

Numero di byte	Valore	Indirizzo	Commento
byte	$00/$80	cia1base + $0E	;(SA) azzera il registro a di controllo ;del CIA 1 e imposta il bit 50/60 Hz
byte	$00/$80	cia2base + $0E	;(SA) azzera il registro a di controllo ;del CIA 2 e imposta il bit 50/60 Hz

Numero di byte	Valore	Indirizzo	Commento
byte	cia2base | #$30|#$04 | GRBANK2	cia2base	;(SA) imposta alcuni bit di gestione ;del bus seriale e seleziona il banco ;di memoria visto dal processore VIC
byte	#$3F	cia2base + $02	;(SA) imposta il registro direzione dati ;del CIA 2
byte	#$7F	cia2base + $0D	;(SA) azzera il registro di controllo ;del CIA 2
		vicBase	;(SA) inizializza il processore VIC ;(SA) posizioni degli sprite
byte	$00,$00	mob0xpos,mob0ypos	;(SA) posizione sprite 0
byte	$00,$00	mob1xpos,mob1ypos	;(SA) posizione sprite 1
byte	$00,$00	mob2xpos,mob2ypos	;(SA) posizione sprite 2
byte	$00,$00	mob3xpos,mob3ypos	;(SA) posizione sprite 3
byte	$00,$00	mob4xpos,mob4ypos	;(SA) posizione sprite 4
byte	$00,$00	mob5xpos,mob5ypos	;(SA) posizione sprite 5
byte	$00,$00	mob6xpos,mob6ypos	;(SA) posizione sprite 6
byte	$00,$00	mob7xpos,mob7ypos	;(SA) posizione sprite 7
byte	$00	msbxpos	;(SA) bit piu' significativo per ;la coordinata x di ogni sprite
byte	st_den|st_25 row|st_bmm|3	grcntrl1	;(SA) registro di controllo della grafica
byte	251	rasreg	;(SA) registro raster di schermo (impostato ;per interrupt al fondo dello schermo)
byte	SKIPFLAG	lpxpos	;(SA) coordinata x della penna ottica ;(flag a sola lettura)
byte	SKIPFLAG	lpypos	;(SA) coordinata y della penna ottica ;(flag a sola lettura)
byte	%00000001	mobenble	;(SA) sprite attivi (solo il mouse)
byte	st_40col	grcntrl2	;(SA) registro di controllo della grafica
byte	%00000000	moby2	;(SA) espansione y degli sprite disabilitata

Numero di byte	Valore	Indirizzo	Commento
byte	(((>COLOR_MATRIX)*2)) & $F0)|(((>SCREEN_BASE)*2) & $0E)	grmemptr	;(SA) puntatore alla memoria grafica
byte	%00001111	grirq	;(SA) registro degli interrupt del processore ;grafico
byte	%00000001	grirqen	;(SA) registro di abilitazione ;degli interrupt del processore grafico
byte	$00	mobprior	;(SA) priorita' degli sprite sullo sfondo
byte	$00	mobmcm	;(SA) opzione multicolor per gli sprite ;disabilitata
byte	%00000000	mobx2	;(SA) espansione x degli sprite disabilitata
Ripristina i Vettori			;(SA) ripristina i vettori del C-64 a $0314 ;copiandoli dalla ROM del Kernel
byte	0	currentMode	;(SA) flag degli stili di visualizzazione ;delle fonti
byte	0	pressFlag	;(SA) nessuna pressione da gestire
byte	%11000000	dispBufferOn	;(SA) visualizza sia sullo schermo principale ;sia sullo schermo nascosto
byte	0	mouseXPos	;(SA) azzera la coordinata x del mouse
byte	0	mouseYPos	;(SA) azzera la coordinata y del mouse
byte	0	mouseOn	;(SA) flag che indica lo stato del mouse, ;dei menu e delle icone
word	mousePicData	msePicPtr	;aggiorna msePicPtr con l'indirizzo dei dati ;grafici del mouse memorizzati a mousePicData
byte	0	windowTop	;(SA) linea superiore per il mascheramento ;dei testi
byte	199	windowBottom	;(SA) linea inferiore per il mascheramento ;dei testi
word	0	leftMargin	;(SA) linea verticale sinistra per ;il mascheramento dei testi
word	319	rightMargin	;(SA) linea verticale destra per ;il mascheramento dei testi
byte	-1	mouseDirection (inputData)	;(SA) nessuna direzione impostata

Numero di byte	Valore	Indirizzo	Commento
word	0	mouseLeft	;(SA) limitazione sinistra di spostamento ;orizzontale del mouse
byte	0	mouseTop	;(SA) limitazione superiore di spostamento ;verticale del mouse
word	SCREEN_PIXEL _WIDTH-1	mouseRight	;(SA) limitazione destra di spostamento ;orizzontale del mouse
byte	SCREEN_PIXEL _HEIGHT-1	mouseBottom	;(SA) limitazione inferiore di spostamento ;verticale del mouse
byte	#MAXIMUM _VELOCITY	maxMouseSpeed	;imposta la velocita' massima ;di spostamento del mouse
byte	#MINIMUM _VELOCITY	minMouseSpeed	;imposta la velocita' minima ;di spostamento del mouse
byte	#MOUSE _ACCELERATION	mouseAccel	;imposta l'accelerazione del mouse
63 byte	dati grafici	mousePicData	;dati grafici del disegno del mouse
1000	#dkgrey*16| ltgrey	COLOR_MATRIX	;riempie lo schermo con il colore ;di fondo ottenuto componendo il grigio ;scuro e il grigio chiaro
byte	#blue	mob0clr	;colore del mouse
byte	#blue	mob1clr	;colore del cursore di input da tastiera
byte	#black	extclr	;imposta il colore del bordo: nero ;copia i dati grafici della freccia ;nello sprite mouse
byte	#$08	interleave	;imposta a 8 l'interspazio su disco ;inizializza il disk drive tramite SetDevice
byte	8	curDrive	
byte	8	curDevice	;nuovamente inizializzato
byte	1	numDrives	;porta a 1 il numero di drive attivi ;imposta l'ora dell'orologio interno
byte	cia1crb & #$7F	cia1crb	;preleva il registro b di controllo ;del CIA 1, attiva l'orologio interno ;e rimemorizza il valore nel registro
byte	#$83	cia1todh	;ora corrente

Numero di byte	Valore	Indirizzo	Commento
byte	0	cialtodmin	;minuti
byte	0	cialtodsec	;secondi
byte	0	cialtod10ths	;1/10 di secondo
byte	0	minutes	;valore in RAM dei minuti
byte	0	seconds	;valore in RAM dei secondi
			;in seguito viene impostata la data corrente
			;in modo che durante gli accessi al disco
			;la data dell'ultimo aggiornamento
			;sia coerente
byte	86	year	;anno attuale
byte	9	month	;mese attuale
byte	20	day	;giorno attuale
byte	12	hour	;ora attuale
			;inizializza i flag e il vettore di allarme
byte	0	alarmSetFlag	
byte	0	alarmCount	
word	0	alarmTmtVector	
			;forza qualsiasi codice turbo correntemente
			;attivo a interrompere la propria esecuzione
			;aggiorna il valore dei principali vettori
			;di sistema
			;inizio di un gruppo (SA)
word	0	appMain	
word	InterruptMain	intTopVector	
word	0	intBotVector	
word	0	mouseVector	
word	0	keyVector	
word	0	inputVector	
word	0	mouseFaultVec	
word	0	otherPressVec	
word	0	stringFaultVec	
word	0	alarmTmtVector	
word	Panic	BRKVector (vettore di gestione dell'istruzione BRK)	

Numero di byte	Valore	Indirizzo	Commento
word	RecoverRectangle	recoverVector	;vettore per ripristinare
			;lo schermo principale
byte	SELECTION_DELAY	selectionFlag	
byte	0	alphaFlag	
byte	ST_FLASH	iconSetFlag	;il valore di default indica il flash
			;dell'icona
byte	0	faultData	
			;fine gruppo (SA)
byte	0	numProcesses	
byte	0	numberAsleep	
byte	0	curIconIndex	

```
;Inizializza i puntatori ai dati grafici degli sprite. In breve, lo spazio visto
;dal chip grafico e' di 16K e quindi occorrono solo 14 bit per indirizzare l'intera area.
;I dati grafici dei disegni per gli sprite occupano 63 byte e iniziano a indirizzi che
;sono multipli interi di 64. Quindi il disegno di ogni sprite e' conservato in memoria
;a un indirizzo i cui 6 bit piu' bassi sono azzerati. Dal momento che i bit per indirizzare
;lo spazio da 16K sono 14, sono sufficienti 8=14-6 bit (un byte) per individuare
;l'indirizzo del disegno di ogni sprite nell'area da 16K
```

Numero di byte	Valore	Indirizzo	Commento
byte	<(spr0pic/64)	spr0pic	
byte	<(spr1pic/64)	spr1pic	
byte	<(spr2pic/64)	spr2pic	
byte	<(spr3pic/64)	spr3pic	
byte	<(spr4pic/64)	spr4pic	
byte	<(spr5pic/64)	spr5pic	
byte	<(spr6pic/64)	spr6pic	
byte	<(spr7pic/64)	spr7pic	

```
disegna lo schermo a griglia                    ;memorizza una matrice grafica a griglia
                                                ;per tutto lo schermo ($A000 - $BF3F)

MoveShortBlock           $FD30, $0314, 32       ;(SA) ripristina i vettori del C-64 che
                                                ;si trovano a pagina 3 della memoria con
                                                ;i valori riportati nella ROM del Kernel

mike 31:
```

21 LE ESPANSIONI RAM E GEOS 128

Introduzione

Nella versione 1.3, GEOS è in grado di gestire in vari modi le espansioni di memoria (REU, Ram Expansion Unit). Questa è una delle caratteristiche che maggiormente differenziano le due versioni, ed è presente anche in GEOS 128. Nella prima parte di questo capitolo esamineremo le operazioni che GEOS compie in modo "trasparente" alle applicazioni e le possibilità d'impiego, da parte dell'applicazione, della RAM aggiuntiva in mansioni parallele a quelle di sistema.

Nella seconda parte del capitolo affronteremo invece il problema della compatibilità di un'applicazione con GEOS 128, e i vari accorgimenti necessari per sfruttare le 80 colonne offerte dal C-128.

Nell'ultima parte del capitolo, infine, illustreremo tutta una serie di piccoli trucchi utili a ogni programmatore. Alcuni sono più che altro espedienti per aggirare i rari bug presenti nel Kernel di GEOS.

Le espansioni RAM

Il C-64, per sua natura, non è in grado di accedere a una quantità di memoria superiore a 64K. Questa limitazione è dovuta alle dimensioni del bus d'indirizzamento della CPU 6510, che, essendo formato da 8 linee distinte, può indirizzare al più 65536 byte (64K). Di fronte a questa limitazione fisica, qualunque incremento di memoria sembra proprio impossibile. Invece l'ostacolo è aggirabile. Alla porta d'espansione (Expansion Port) del C-64 arrivano diverse linee, fra le quali sono presenti l'intero bus indirizzi, il bus dati e una linea che consente di disabilitare temporaneamente la CPU. È quindi possibile che un processore esterno possa impadronirsi temporaneamente del

computer ed eseguire operazioni direttamente nella memoria del C-64. Le espansioni di memoria sfruttano proprio questa possibilità. Sono infatti dotate di un processore interno in grado di eseguire poche operazioni ad altissima velocità, con grandi quantità di dati. La CPU del C-64 non può quindi accedere direttamente ai banchi di memoria contenuti nell'espansione, ma può comunicare con il processore esterno, passandogli alcuni parametri e ordinandogli di compiere alcune operazioni. Nel momento in cui il processore dell'espansione riceve il comando, disabilita il 6510 ed esegue le operazioni richieste interagendo con la memoria del computer e quella dell'espansione. I banchi sono tutti da 64K e la dimensione dell'espansione dipende dal numero di banchi di cui è formata.

Per comunicare con l'espansione il 6510 deve fornire alcuni parametri: il banco di memoria con il quale avviene la comunicazione, l'indirizzo all'interno del banco e quello all'interno del C-64 da cui deve iniziare l'operazione, e il numero di byte interessati. Questi parametri devono essere memorizzati in particolari registri dell'espansione, allocati da EXP_BASE ($DF00) in poi. Di solito in quest'area del campo indirizzabile non è presente alcuna memoria. Con l'inserimento dell'espansione diventano accessibili i registri di controllo del processore esterno. Quando i parametri sono stati impostati, il 6510 deve memorizzare nel registro comandi le operazioni assegnate al processore esterno. A questo punto, ogni volta che viene impartito un comando, il processore dell'espansione lo esegue disabilitando temporaneamente il 6510. Il 6510 riprende il controllo solo a operazione compiuta, e non vi partecipa in nessun modo. L'operazione, quindi, avviene in modo del tutto "trasparente" per quanto riguarda il C-64.

Le principali operazioni che si possono realizzare grazie all'espansione sono quattro. Ciascuna richiede un diverso comando. Il comando **VERIFY** consente di confrontare blocchi di dati delle stesse dimensioni, contenuti rispettivamente nella memoria del C-64 e in quella dell'espansione. Il comando **STASH** permette il trasferimento di un blocco di dati dalla memoria del C-64 all'espansione. Il comando **FETCH**, viceversa, trasferisce un blocco di dati dall'espansione alla memoria del C-64. **SWAP** infine permette di scambiare simultaneamente un blocco di dati in memoria con un blocco delle stesse dimensioni contenuto nell'espansione.

Ovviamente, la quantità di memoria coivolta in ciascuna operazione non può superare le dimensioni del banco di memoria con cui si lavora. La velocità del trasferimento di dati raggiunge i 200K al secondo, e questo rende conveniente utilizzare le espansioni RAM anche solo per spostare ingenti quantità di dati da un'area all'altra della memoria del computer.

L'ultima caratteristica importante ai fini della gestione delle espansioni in ambiente GEOS riguarda il reset del computer. Contrariamente a quanto si potrebbe pensare, le espansioni RAM non vengono coinvolte nel reset del computer e le informazioni in esse immagazzinate restano inalterate. Solo spegnendo il computer o cancellandole volontariamente è possibile perderne il contenuto.

GEOS e le espansioni

Ora che sappiamo qualcosa di più sul funzionamento delle espansioni, siamo in grado d'illustrare come vengono impiegate dal Kernel di GEOS e quali configurazioni si possono ottenere. Il Kernel è in grado di "vedere" espansioni fino a 512K di memoria. Per essere più precisi, può interagire con ogni memoria esterna di dimensioni complessive non superiori ai 512K e organizzata in banchi da 64K. Le possibili quantità sono quindi 64K, 128K, 192K, 256K, 320K, 384K, 448K, 512K. Le azioni che può svolgere il Kernel dipendono anche dalla quantità di memoria esterna disponibile.

L'utente sceglie il tipo di configurazione più conveniente alle sue esigenze attraverso l'applicazione Configure, che riconosce il tipo di espansione inserita e (a seconda della quantità di memoria aggiuntiva disponibile) offre all'utente diverse possibili configurazioni di sistema.

Vi sono due operazioni che il Kernel può compiere sempre, anche con la più piccola espansione: muovere in maniera rapidissima aree di dati da un punto all'altro della memoria, e memorizzare i codici necessari per installare nuovamente il sistema senza che vengano compiuti accessi al disco.

Le applicazioni che devono spostare grandi quantità di dati, come per esempio geoPaint quando muove la finestra di lavoro sul foglio da disegno, impiegano spesso la routine MoveData del Kernel di GEOS. Ma MoveData è molto lenta quando deve eseguire spostamenti consistenti, dal momento che deve ricorrere a un loop di istruzioni. Se invece è presente un'espansione, si può delegare questo compito al processore esterno: il Kernel non fa altro che trasferire nell'espansione l'area interessata, e subito dopo la ritrasferisce nel computer al nuovo indirizzo. Il tempo complessivo necessario all'operazione è molto inferiore a quello richiesto dal tradizionale loop di MoveData. Quando si sceglie questa opzione, chiamata anch'essa **MoveData**, Configure altera il sistema opportunamente in modo che MoveData esegua le sue funzioni impiegando l'espansione RAM.

La seconda operazione che il Kernel è in grado di compiere con un'espansione consiste nel trasferimento dell'intero sistema e dei codici necessari nell'espansione, in modo che sia possibile riattivarlo senza accedere al disco. Con questa opzione, quando l'utente ordina al Kernel di cedere il controllo al Basic, l'intero Kernel viene trasferito nell'espansione insieme a un programma di caricamento. Per rientrare in ambiente GEOS, l'utente può premere il tasto "restore", oppure dare la sys 49152, o infine eseguire il file Rboot; l'intero Kernel viene così trasferito dall'espansione in memoria in meno di un secondo e gli viene subito ceduto il controllo. A questo punto il Kernel provvede a caricare e mandare in esecuzione deskTop.

L'opzione ora descritta, che Configure identifica come **RAM Reboot**, è particolarmente utile quando si devono mandare in esecuzione molti file non GEOS compatibili, rientrando ogni volta in ambiente GEOS nel tempo più breve possibile. Al rientro, la configurazione precedente viene mantenuta, compreso il contenuto dell'eventuale

RAM disk, di cui parleremo fra poco. Si noti che se il Kernel sta simulando anche un RAM disk sull'espansione, e nel RAM disk risiede una copia di deskTop, quando il sistema viene riattivato dall'espansione anche deskTop viene caricata dall'espansione.

MoveData e RAM Reboot possono essere selezionate simultaneamente e non interferiscono con gli altri possibili impieghi dell'espansione. Se la quantità di memoria esterna disponibile supera i 256K, GEOS è in grado di sfruttarla per realizzare un drive di tipo Shadowed o RAM disk. Naturalmente queste sono opzioni alternative l'una all'altra. Il nuovo disk drive 1541 "virtuale" che si viene a creare può essere sia il drive A sia il drive B.

Il drive di tipo **Shadowed** è un disk drive 1541 reale affiancato a un buffer della stessa capacità del disco formattato. Ogni volta che l'utente carica un'applicazione o un file dati in memoria, il file viene trasferito nel buffer in modo che il Kernel possa caricarlo dal buffer (e non da disco) in tempi brevissimi, nel caso che se ne presenti la necessità. Ogni volta che un'applicazione salva un file su disco, il file viene copiato anche nel buffer. In questo modo il caricamento di tutti i file letti o salvati almeno una volta può avvenire direttamente dall'espansione.

In alternativa al disco Shadowed, l'utente può configurare il Kernel di GEOS in modo che impieghi l'espansione RAM come un disco 1541 virtuale, ovvero come un **RAM disk**. Il disco virtuale viene identificato come drive A se il drive reale è il drive B, e viceversa. Per le applicazioni e per l'utente, è come se fosse collegato un secondo disk drive 1541. La differenza è che i file salvati sul disco virtuale vengono caricati in tempi brevissimi (il tempo di premere due volte il pulsante del mouse sull'icona), e i salvataggi dei dati su disco virtuale sono istantanei. Bisogna però ricordare che il contenuto del RAM disk viene completamente perso se si spegne il computer. Al contrario, rimane inalterato se si esegue un reset del computer e successivamente si ricarica GEOS. Dal momento che le due opzioni non possono convivere, l'utente deve decidere quale può essergli più utile quando effettua le sue scelte tramite Configure.

L'applicazione Configure è di tipo AUTO_EXEC, e quindi durante il boot del sistema (l'installazione) viene eseguita sempre prima di deskTop. Quando riceve il controllo, Configure verifica il contenuto della locazione firstBoot, e se è $00 rileva che deskTop non è stata caricata nemmeno una volta e quindi che è in corso l'installazione. Di conseguenza Configure non è stata chiamata dall'utente, ma dal sistema. Provvede allora a configurare automaticamente il sistema secondo le specifiche che sono state salvate dall'utente la volta precedente, o predispone quelle di default.

Quando invece Configure viene chiamata dall'utente, si accorge che il contenuto di firstBoot è diverso da $00 e decide quindi che l'utente deve ricevere il controllo per introdurre una nuova configurazione di sistema, che verrà salvata su disco. Da questo momento, quando l'applicazione verrà eseguita automaticamente dal sistema utilizzerà i dati salvati su disco per configurare il sistema secondo quanto stabilito dall'utente.

Tutte le possibilità operative ora descritte, offerte dal Kernel di GEOS V1.3, sono completamente "trasparenti" per le applicazioni. Queste infatti non sono tenute a sapere se il drive B è virtuale, o se il drive A è di tipo Shadowed, dal momento che il sistema maschera ogni differenza, e neppure se la routine MoveData impiega il processore dell'espansione o no. Le applicazioni continuano a usare le routine del Kernel come è sempre avvenuto, cioè verificando esclusivamente se sono presenti due disk drive o uno solo.

Le applicazioni e le espansioni

Anche se GEOS è in grado di gestire efficientemente e in maniera autonoma l'eventuale espansione RAM, può accadere che un'applicazione desideri impiegarne un banco o più per svolgere compiti diversi. Per esempio, memorizzare le fonti carattere senza che l'espansione sia necessariamente impiegata come RAM disk. A questo scopo, GEOS mette a disposizione delle applicazioni cinque routine di sistema appositamente realizzate per impartire comandi all'espansione di memoria. Le applicazioni possono accedere alla variabile ramExpSize per determinare il numero di banchi da 64K di cui è composta l'espansione correntemente inserita. Gli indirizzi all'interno di ogni banco sono relativi all'inizio del banco stesso, e prescindono quindi dal suo numero d'ordine. Ricordiamo infine che queste routine sono disponibili solo nella versione 1.3 di GEOS e successive, e in GEOS 128.

StashRAM

Funzione: Trasferisce un'area di dati dalla memoria del C-64 a uno dei banchi dell'espansione, all'indirizzo desiderato.

Versione: GEOS V1.3 e successive, GEOS 128

Indirizzo: $C2C8

Parametri:
- r0 indirizzo all'interno del C-64
- r1 indirizzo all'interno del banco dell'espansione
- r2 numero di byte interessati al trasferimento
- r3L banco all'interno dell'espansione (numerati da 0 a 7)

Restituisce: x codice d'errore

0	l'operazione ha avuto successo
DEV_NOT_FOUND	numero di banco non valido, o espansione non presente

Distrugge: a, y

Sinossi: Questa routine permette alle applicazioni di trasferire nel banco r3L dell'espansione, e al suo indirizzo interno r1, l'area di memoria che inizia all'indirizzo r0 del C-64 ed è composta da r2 byte.

FetchRAM

Funzione: Questa routine permette alle applicazioni di trasferire una particolare area di memoria (contenuta in un banco dell'espansione) all'interno della memoria del C-64, all'indirizzo prestabilito.

Versione: GEOS V1.3 e successive, GEOS 128

Indirizzo: $C2CB

Parametri:

r0	indirizzo all'interno del C-64
r1	indirizzo all'interno del banco dell'espansione
r2	numero di byte interessati al trasferimento
r3L	banco all'interno dell'espansione (numerati da 0 a 7)

Restituisce:

x	codice d'errore	
	0	l'operazione ha avuto successo
	DEV_NOT_FOUND	numero di banco non valido, o espansione non presente

Distrugge: a, y

Sinossi: FetchRAM permette di trasferire il blocco di dati memorizzato nel banco r3L all'indirizzo interno r1, e composto da r2 byte, nella memoria del computer all'indirizzo r0.

SwapRAM

Funzione: SwapRAM consente alle applicazioni di scambiare tra loro un'area della memoria e un'area di uguale dimensione presente nell'espansione.

Versione: GEOS V1.3 e successive, GEOS 128

Indirizzo: $C2CE

Parametri:

r0	indirizzo all'interno del C-64
r1	indirizzo all'interno del banco dell'espansione
r2	numero di byte interessati allo scambio
r3L	banco all'interno dell'espansione (numerati da 0 a 7)

Restituisce:

x	codice d'errore	
	0	l'operazione ha avuto successo
	DEV_NOT_FOUND	numero di banco non valido, o espansione non presente

Distrugge: a, y

Sinossi: Tramite SwapRAM le applicazioni possono scambiare l'area della memoria del C-64 che inizia a r0, con l'area del banco r3L dell'espansione che inizia all'indirizzo r1. Il numero di byte interessato allo scambio è r2.

VerifyRAM

Funzione: VerifyRAM consente di confrontare due aree di memoria, una contenuta nel C-64 e una nell'espansione.

Versione: GEOS V1.3 e successive, GEOS 128

Indirizzo: $C2D1

Parametri:
- r0 indirizzo all'interno del C-64
- r1 indirizzo all'interno del banco dell'espansione
- r2 numero di byte interessati al confronto
- r3L banco all'interno dell'espansione (numerati da 0 a 7)

Restituisce: x codice d'errore

0	l'operazione ha avuto successo
DEV_NOT_FOUND	numero di banco non valido, o espansione non presente

a byte di stato dell'espansione a cui è stata applicata l'operazione logica AND con il valore $60, in modo da azzerare tutti i bit tranne il 5 e il 6
- $40 il confronto ha avuto successo
- $20 il confronto non ha avuto successo

Distrugge: y

Sinossi: VerifyRAM permette di confrontare l'area di memoria del C-64 che inizia a r0 con l'area contenuta nel banco r3L dell'espansione all'indirizzo interno r1. Il numero di byte coinvolti nel confronto è r2.

DoRAMOp

Funzione: Consente alle applicazioni di impartire al processore dell'espansione un particolare comando (è una routine di livello molto basso).

Versione: GEOS V1.3 e successive, GEOS 128

Indirizzo: $C2D4

Parametri:

r0	indirizzo all'interno del C-64
r1	indirizzo all'interno del banco dell'espansione
r2	numero di byte interessati all'operazione
r3L	banco all'interno dell'espansione (numerati da 0 a 7)
y	comando che dev'essere impartito. Per esempio: STASH= %10010000 FETCH= %10010001 VERIF = %10010011 SWAP = %10010010

Restituisce:

x	codice d'errore	
	0	l'operazione ha avuto successo
	DEV_NOT_FOUND	numero di banco non valido, o espansione non presente

Distrugge: a, y

Sinossi: Tramite DoRAMOp, le routine possono accedere direttamente al registro comandi dell'espansione per impartire comandi diversi da quelli riconosciuti dal Kernel di GEOS. Il comando dev'essere memorizzato in y.

Le applicazioni e la compatibilità con GEOS 128

GEOS 128 si può considerare un parente molto stretto di GEOS V1.3 per il C-64. Tutte le routine riportate nella jump table di sistema di GEOS V1.3 sono fedelmente riportate in GEOS 128, e i parametri non sono diversi. Anche tutte le variabili di sistema disponibili in GEOS V1.3 sono le stesse di GEOS 128. Per queste ragioni *quasi* tutte le applicazioni prodotte dalla Berkeley Softworks per GEOS 64 possono essere eseguite in ambiente GEOS 128. Il "quasi" è necessario perché esiste pur sempre qualche differenza. Le applicazioni che devono accedere al Kernel originale del computer non sono compatibili con GEOS 128, per via delle sostanziali differenze presenti fra il Kernel del C-128 e quello del C-64. Fra queste applicazioni rientrano, per esempio, il desk accessory calculator e geoCalc, le quali svolgono operazioni matematiche complesse accedendo alle routine matematiche del computer. Se si desidera che l'applicazione sia comunque compatibile con entrambi i sistemi, è necessario realizzare due distinte tavole di salto, una per ogni Kernel.

GEOS 128, oltre a riprodurre quasi fedelmente le caratteristiche di GEOS V1.3 per il C-64, ne possiede diverse, fra cui la grafica a 80 colonne. Vediamo quali sono i passi necessari affinché le applicazioni che sono state create esplicitamente per GEOS 64 possano impiegare lo schermo a 80 colonne del C-128.

La grafica a 80 colonne con GEOS 128

Se si desidera che l'applicazione sia in grado di abilitare e gestire lo schermo a 80 colonne offerto da GEOS 128, si devono seguire alcune direttive fondamentali.

1) Per prima cosa è necessario che GEOS 128, caricando il file, sia in grado di determinare se è compatibile con il modo a 80 colonne. GEOS 128 ha bisogno di questa informazione perché se il modo a 80 colonne è già abilitato l'applicazione non può accedere allo schermo, ed è necessario avvisare l'utente o tornare automaticamente nel modo a 40 colonne. Quando GEOS 128 carica un file in memoria, accede alla locazione individuata dalla costante OFF_128_FLAGS ($96) nel File Header del file. In GEOS 64 questa locazione rientra nello spazio allocato per il nome permanente ed è sempre azzerata. In ambiente GEOS 128 questa locazione assume un particolare

significato. Vediamo i flag di cui è composta:

Bit 7	Bit 6	Significato
0	0	L'applicazione impiega solo lo schermo a 40 colonne
0	1	L'applicazione impiega sia lo schermo a 40 colonne sia quello a 80 colonne
1	0	L'applicazione non può essere eseguita in ambiente GEOS 128
1	1	L'applicazione impiega esclusivamente lo schermo a 80 colonne

Le routine di caricamento come LdApplic e LdDeskAcc restituiscono l'errore INCOMPATIBLE se questi flag indicano che l'applicazione non può essere eseguita nel corrente modo di visualizzazione. In futuro forse anche la routine FindFTypes si evolverà fino a poter distinguere i file in base agli stessi criteri.

L'applicazione deve allora riportare il valore $40 nella locazione OFF_128_FLAGS del suo File Header. In questo modo GEOS 128 consente entrambi i modi grafici (40 e 80 colonne).

2) Nel modo a 80 colonne è necessario allargare tutti i menu in modo che siano in grado di contenere la fonte di sistema, che è più larga. La consuetudine della Berkeley è quella di impostare il valore del limite destro nelle strutture dei menu sulla base del valore contenuto in graphicsMode ($80 per 80 colonne, $00 per 40 colonne). La variabile graphicsMode è prevista solo da GEOS 128 ed è allocata all'indirizzo $003F.

3) La maggior parte dei cambiamenti nei valori grafici necessari per la compatibilità con il modo a 80 colonne possono essere realizzati impostando a 1 il bit 15 di tutte le coordinate x e di tutte le larghezze che vengono passate al sistema. Nel modo a 40 colonne questo viene ignorato, nel modo a 80 colonne serve per adeguare tutte le dimensioni orizzontali. Così facendo l'immagine ha sempre le stesse dimensioni sullo schermo. Per esempio, la coordinata x = 50 pixel in modo a 40 colonne, dev'essere passata a GEOS 128 nella forma $8032, in maniera che a 80 colonne diventi $0064 (100 pixel).

4) Nel menu geos dell'applicazione (o in ogni caso all'interno di qualsiasi menu) dev'essere disponibile la voce "switch 40/80". La routine associata all'evento deve semplicemente eseguire l'operazione logica EOR fra graphicsMode e la costante $80 (inverte il valore del bit 7), memorizzare il risultato in graphicsMode e chiamare la routine SetNewMode ($C2DD solo GEOS 128). In seguito l'applicazione deve preoccuparsi di ridisegnare lo schermo corrente nel nuovo modo grafico. Se le dimensioni orizzontali hanno già il bit 15 impostato a 1, la routine che ridisegna lo

schermo non deve subire variazioni. Ecco un esempio dei codici associati alla voce switch 40/80:

```
SwitchDsp:
   lda   #$80
   eor   graphicsMode
   sta   graphicsMode
   jsr   SetNewMode             ;$C2DD questa routine e' disponibile solo in GEOS 128
   ...                          ;codici per inizializzare nuovamente lo schermo
```

5) L'accorgimento adottato per adeguare le dimensioni orizzontali alle 80 colonne (bit 15 impostato a 1) non sempre si rivela efficace. Quando il valore di una coordinata orizzontale viene raddoppiato per il modo a 80 colonne, inevitabilmente il bit 0 della word risultante è sempre azzerato. In alcuni casi questa può essere una grave limitazione: per esempio quando si desidera riempire lo schermo con una matrice grafica che si estenda fino al lato destro. Per risolvere questo problema sono state introdotte alcune variazioni nelle routine grafiche di GEOS 128: il bit 15 della word continua ad avere lo stesso significato (se è impostato a 1 fornisce l'adeguamento delle dimensioni al modo in 80 colonne), mentre il bit 14 dà una nuova informazione, ma solo nel modo in 80 colonne. Il bit 14 diventa il bit 0 della word risultante dall'operazione di "raddoppiamento". Se per esempio si desidera individuare il lato destro dello schermo, la coordinata orizzontale dev'essere $C000 + 319.

Grazie a questi cinque accorgimenti si dovrebbe riuscire a sfruttare agevolmente il modo a 80 colonne del C-128. In ogni caso è possibile che siano necessari alcuni ritocchi nella fase di sperimentazione dei layout dell'applicazione (questo tipo di verifica è sempre consigliabile).

I piccoli trucchi del mestiere

In quest'ultimo paragrafo riportiamo una serie di piccoli accorgimenti di cui il programmatore dovrebbe tener conto nella realizzazione delle applicazioni. In qualche caso si tratta di aggirare i piccoli bug ancora presenti nella struttura di GEOS.

1) Se l'applicazione può essere eseguita in ambiente GEOS 128, ed è in grado di gestire il secondo drive, si presti una particolare attenzione a tutte le chiamate della routine PutDirHead, e ogni volta si inserisca immediatamente prima l'istruzione jsr EnterTurbo. Questo accorgimento è necessario perché nella prima produzione di GEOS 128 è presente un bug nel driver per il 1571: manca la chiamata a EnterTurbo. Il risultato è che in certe circostanze la chiamata a PutDirHead può anche rovinare il disco. Questo accorgimento non crea incompatibilità con GEOS 64. Nella prossima produzione di GEOS 128 il bug verrà eliminato.

2) Se il programma è in grado di eseguire i desk accessory, si ricordi che tutti i programmi Blackjack fino a oggi venduti alterano il contenuto della word a $4C95 che non si trova nell'area temporaneamente salvata su disco. Per porre rimedio a questo bug, è necessario che il codice responsabile dei desk accessory sia rispettivamente preceduto e seguito dalle istruzioni PushW $4C95 e PopW $4C95. Inoltre GEOS 64 non salva moby2 durante l'esecuzione dei desk accessory. Questo significa che gli sprite possono facilmente essere ingranditi in altezza dai DA e quindi modificati. Ecco un esempio pratico di come agire, sia sull'applicazione sia sul desk accessory:

Applicazioni:

```
    ldx     CPU_DATA               ;salva lo stato di moby2 sullo stack
    LoadB   CPU_DATA, #IO_IN
    PushB   moby2
    stx     CPU_DATA

    PushW   $4C95                  ;salva il dato che BJ distruggerebbe

CARICARE IL DESK ACCESSORY QUI

    PopW    $4C95                  ;ripristina la word
    ldx     CPU_DATA               ;ripristina moby2
    LoadB   CPU_DATA, #IO_IN
    PopB    moby2
    stx     CPU_DATA
```

Desk accessory:

```
InitCode:
    ldx     CPU_DATA               ;salva lo stato di moby2
    LoadB   CPU_DATA, #IO_IN
    lda     moby2
    sta     savedmoby2
    lda     #$XX                   ;imposta moby2 secondo le esigenze
    sta     moby2
    stx     CPU_DATA

ExitCode:
    ldx     CPU_DATA               ;ripristina lo stato di moby2
    LoadB   CPU_DATA, #IO_IN
    lda     savedmoby2
    sta     moby2
    stx     CPU_DATA
```

3) GEOS potrebbe non funzionare correttamente se nessuna icona è stata definita. Se l'applicazione non impiega icone, è meglio definirne una fittizia per evitare problemi. Si può definirla in modo che sia alta una linea di scansione, larga un byte e con il puntatore ai dati grafici azzerato.

4) In ambiente GEOS 128, non si deve mai assumere che il concatenamento dei blocchi della directory inizi con il settore $12/$01, o che il Directory Header Block sia situato all'indirizzo T/S $12/$00, perché per il 1581 il formato è diverso. Si devono sempre eseguire le routine GetDirHead, PutDirHead, Get1stDirEntry e GetNextDirEntry presenti nel disk driver corrente.

5) Non si deve mai cambiare con un intervento diretto il contenuto della locazione curDrive ($BA). Bisogna invece chiamare SetDevice per indirizzare il disk drive desiderato.

6) Nei desk accessory: il codice d'inizializzazione dei desk accessory potrebbe, in particolari condizioni, decidere che il programma non può "girare" e quindi restituire subito il controllo al sistema. Questa operazione di "quit" dal codice d'inizializzazione non può essere realizzata direttamente. È necessario che il codice aggiorni il vettore appMain con l'indirizzo di una routine interna e restituisca il controllo. Dovrà essere poi questa routine a eseguire la procedura di chiusura del desk accessory.

7) Nei box di dialogo: il comando DB_USR_ROUT viene eseguito prima che siano state disegnate le icone. Se la routine personalizzata deve disegnare qualcosa sopra le icone, deve impostare il vettore appMain con l'indirizzo di un'altra routine, e delegarla per visualizzare i disegni sopra le icone.

8) Mai usare la routine MoveData per muovere il contenuto dei registri r0 - r15.

9) I box di dialogo possono gestire non più di otto icone contemporaneamente. Se il box deve visualizzare più di otto icone, deve gestirle autonomamente attraverso il vettore otherPressVec.

10) Si ricordi che la gestione degli eventi (processi, routine puntate da keyVector e appMain) è attiva durante il periodo in cui un menu è aperto. La routine puntata da otherPressVec è attiva in maniera parziale: analizza solo i rilasci del pulsante. Se l'applicazione desidera ignorare questi eventi durante l'apertura di un menu (situazione molto frequente) non si deve dimenticare di disabilitarli.

11) Le chiamate a DoMenu e DoIcon muovono il mouse. Dal momento che in genere questo non è desiderabile, si deve agire come segue:

```
PushW    mouseXPos
```

```
PushB    mouseYPos
jsr      DoIcons              ;o DoMenu
PopB     mouseYPos
PopW     mouseXPos
```

12) Se l'applicazione interagisce con recoverVector (per ripristinare lo sfondo coperto da un menu o da un box di dialogo) si ricordi che la routine individuata dal vettore viene chiamata due volte quando si ripristina lo sfondo sottostante a un box di dialogo.

13) I codici di interrupt di GEOS non impostano a 1 il bit del modo decimale nel PSW. Dal momento che i conteggi vengono svolti con questo bit azzerato, le chiamate di interrupt non devono mai avvenire mentre è attivato il modo decimale.

14) Se l'applicazione disattiva (blank) lo schermo oppure scrive in grcntr11 ($D011), ci si assicuri che il bit 7 sia sempre a 0. Dal momento che accidentalmente, nel corso di diverse operazioni, questo bit può diventare 1, si possono impiegare i seguenti codici per azzerarlo:

```
lda      grcntr11             ;preleva il valore corrente
...                           ;ne elabora i bit
and      #%01111111           ;reset bit 7
sta      grcntr11             ;memorizza il nuovo valore
```

15) Il modulo di gestione dei menu residente nel Kernel di GEOS, quando viene attivato dalla routine DoMenu, memorizza nel vettore mouseFaultVec l'indirizzo di una routine interna che controlla la chiusura del menu corrente quando il mouse ne oltrepassa i bordi. Questo impiego del vettore mouseFaultVec, quando è attiva una una struttura di menu, può entrare in conflitto con l'applicazione qualora questa desideri usare mouseFaultVec per altri scopi. La soluzione al problema si ottiene con due interventi, uno nella routine d'inizializzazione dell'applicazione, uno nella routine di servizio che l'applicazione desidera assegnare al vettore mouseFaultVec.

Primo intervento. Quando l'applicazione vuole usare mouseFaultVec e contemporaneamente una struttura di menu, la routine d'inizializzazione deve, dopo la chiamata a DoMenu, memorizzare in un vettore interno il contenuto del vettore mouseFaultVec. Compiuta questa operazione, può memorizzare in mouseFaultVec l'indirizzo della routine di servizio che si desidera assegnare al vettore.

Secondo intervento. La routine di servizio associata a mouseFaultVec, quando riceve il controllo, deve verificare se la propria esecuzione è stata richiesta dal sistema in quanto il mouse ha oltrepassato uno dei limiti posti dall'applicazione. In caso affermativo, può compiere le sue funzioni e restituire il controllo a MainLoop con l'istruzione rts. In caso contrario, deve cedere il controllo alla routine il cui indirizzo è stato memorizzato nel vettore interno dalla routine d'inizializzazione.

APPENDICE A: COSTANTI

```
;*******************************************************************************************
;
;                                    geosConstants
;
;     Questo file contiene le costanti da impiegare nelle applicazioni GEOS compatibili
;
;
;
;*******************************************************************************************

TRUE                    =    -1
FALSE                   =     0

;Le costanti che seguono vengono utilizzate nel registro CPU_DATA ($0001 per il C-64,
;$FF00 per il C-128) per impostare la configurazione RAM/ROM della mappa di memoria

IO_IN                   = $35              ;60K RAM, 4K di spazio I/O attivati
RAM_64K                 = $30              ;64K RAM
KRNL_BAS_IO_IN          = $37              ;ROM del Kernel, del Basic e spazio di I/O
                                           ;attivati
KRNL_IO_IN              = $36              ;ROM del Kernel e spazio di I/O attivati

;per il C-128

CIO_IN                  = $7E              ;60K RAM, 4K di spazio I/O attivati
CRAM_64K                = $7F              ;64K RAM
```

```
CKRNL_BAS_IO_IN          = $40              ;Kernel, I/O e Basic attivati
CKRNL_IO_IN              = $4E              ;Kernel e I/O attivati


; ******************************************************************************
;                               Costanti Generali
; ******************************************************************************


PROMPT_DELAY             =     60           ;Il valore massimo e' 63
VERT_SPACE               =     2
HORI_SPACE               =     4

POSITIVE                 =     %00000000
NEGATIVE                 =     %10000000


; ******************************************************************************
;                                    Menu
; ******************************************************************************


MAX_ME_ITEMS             =     15
MAX_ME_NESTING           =     4
DATA_BUFFER_SIZE         =     8
PUTCHAR_BUFFER_SIZE      =     8

HORIZONTAL               =     %00000000
VERTICAL                 =     %10000000
CONSTRAINED              =     %01000000
UN_CONSTRAINED           =     %00000000

;Ritardo (delay) fra le due inversioni dell'opzione selezionata di un menu

SELECTION_DELAY          =     10           ;1/6 di secondo

;Offset per le variabili all'interno della struttura di definizione dei menu

OFF_MY_TOP               =     0            ;Offset alla coordinata y del lato superiore
                                            ;del menu
OFF_MY_BOT               =     1            ;Offset alla coordinata y del lato inferiore
                                            ;del menu
```

```
OFF_MX_LEFT            =    2             ;Offset alla coordinata x del lato sinistro
                                          ;del menu
OFF_MX_RIGHT           =    4             ;Offset alla coordinata x del lato destro
                                          ;del menu
OFF_NUM_M_ITEMS        =    6             ;Offset all'indicatore
                                          ;di allineamento|movimento|numero di voci
OFF_1ST_M_ITEM         =    7             ;Offset al primo gruppo di dati
                                          ;di definizione della prima voce

SUB_MENU               =    $80           ;Queste costanti vengono utilizzate per
                                          ;indicare se la singola voce del menu
DYN_SUB_MENU           =    $40           ;causa un evento o l'apertura
                                          ;di un sotto-menu
MENU_ACTION            =    $00

; ****************************************************************************************
;                                      Processi
; ****************************************************************************************

MAX_PROCESSES          =    20            ;Questo e' il massimo valore che puo'
                                          ;contenere la variabile numProcesses
                                          ;allocata all'indirizzo $877D
SLEEP_MAX              =    20

;Valori possibili per le variabili allocate da processFlags ($8719) in poi, non
;direttamente accessibili per le applicazioni

SET_RUNABLE            =    %10000000     ;Flag processo eseguibile
SET_BLOCKED            =    %01000000     ;Flag processo bloccato
SET_FROZEN             =    %00100000     ;Flag processo congelato
SET_NOTIMER            =    %00010000     ;Flag processo non temporizzato

RUNABLE_BIT            =    7             ;Bit del Flag eseguibile
BLOCKED_BIT            =    6             ;Bit del Flag bloccato
FROZEN_BIT             =    5             ;Bit del Flag congelato
NOTIMER_BIT            =    4             ;Bit del Flag non temporizzato

; ****************************************************************************************
;                                      Testo
; ****************************************************************************************

;Costanti per la variabile currentMode
```

```
SET_UNDERLINE       =   %10000000
SET_BOLD            =   %01000000
SET_REVERSE         =   %00100000
SET_ITALIC          =   %00010000
SET_OUTLINE         =   %00001000
SET_SUPERSCRIPT     =   %00000100
SET_SUBSCRIPT       =   %00000010
SET_PLAINTEXT       =   %00000000

UNDERLINE_BIT       =   7
BOLD_BIT            =   6
REVERSE_BIT         =   5
ITALIC_BIT          =   4
OUTLINE_BIT         =   3
SUPERSCRIPT_BIT     =   2
SUBSCRIPT_BIT       =   1

;Costanti per la routine PutString le costanti indicate con un asterisco si possono
;impiegare anche con PutChar

EOF                 =   0           ;Fine del testo
NULL                =   0           ;Fine della stringa
BACKSPACE           =   8           ;* muove a sinistra di uno spazio carattere
TAB                 =   9           ;Tabulatore
FORWARDSPACE        =   9           ;* muove a destra di uno spazio
LF                  =   10          ;* muove verso il basso di una riga
HOME                =   11          ;* muove all'angolo superiore sinistro dello
                                    ;schermo
UPLINE              =   12          ;* muove verso l'alto di una riga
PAGE_BREAK          =   12          ;Fine pagina
CR                  =   13          ;* muove all'inizio della riga successiva
ULINEON             =   14          ;* attiva lo stile sottolineato
ULINEOFF            =   15          ;* disattiva lo stile sottolineato
ESC_GRAPHICS        =   16          ;Codice di controllo per la stringa grafica
ESC_RULER           =   17          ;Codice di controllo per la riga
                                    ;di definizione
REV_ON              =   18          ;* attiva l'output su schermo in negativo
REV_OFF             =   19          ;* disattiva l'output su schermo in negativo
GOTOX               =   20          ;Utilizza la word successiva come nuova
                                    ;coordinata x del cursore
GOTOY               =   21          ;Utilizza il byte successivo come nuova
                                    ;coordinata y del cursore
```

```
GOTOXY              =   22          ;Utilizza la word e il byte successivi
                                    ;come nuove coordinate x e y del cursore
NEWCARDSET          =   23          ;La word che segue indica il FontID, e
                                    ;il byte successivo lo stile
BOLDON              =   24          ;* attiva lo stile nero
ITALICON            =   25          ;* attiva lo stile corsivo
OUTLINEON           =   26          ;* attiva lo stile outline
PLAINTEXT           =   27          ;* attiva lo stile tondo
USELAST             =   127         ;Cancella carattere
SHORTCUT            =   128         ;Carattere shortcut (il carattere Commodore)


; ******************************************************************************************
;                                         Tastiera
; ******************************************************************************************


KEY_QUEUE_SIZE      =   16          ;Dimensione della coda (buffer) di tastiera
KEY_REPEAT_COUNT    =   15          ;1/4 di secondo: tempo di auto-repeat
                                    ;per la tastiera (massimo 254 e non 255)

KEY_INVALID         =   31
KEY_F1              =   1
KEY_F2              =   2
KEY_F3              =   3
KEY_F4              =   4
KEY_F5              =   5
KEY_F6              =   6
KEY_F7              =   14
KEY_F8              =   15
KEY_UP              =   16
KEY_DOWN            =   17
KEY_HOME            =   18
KEY_CLEAR           =   19
KEY_LARROW          =   20
KEY_UPARROW         =   21
KEY_STOP            =   22
KEY_RUN             =   23
KEY_BPS             =   24

.ifdef              C-128
.if                 C-128
```

```
KEY_HELP                =     25
KEY_ALT                 =     26
KEY_ESC                 =     27
KEY_NOSCRL              =     7
KEY_ENTER               =     11
.endif
.endif

KEY_LEFT                =     BACKSPACE
KEY_RIGHT               =     30
KEY_DELETE              =     29
KEY_INSERT              =     28


; ********************************************************************************************
;                                           Mouse
; ********************************************************************************************


MOUSE_ACCELERATION      =     127          ;Accelerazione del mouse
MAX_VELOCITY            =     127          ;Velocita' massima del mouse
MIN_VELOCITY            =     30           ;Velocita' minima del mouse
SET_MSE_ON              =     %10000000
SET_MENUON              =     %01000000
SET_ICONSON             =     %00100000

MOUSEON_BIT             =     7
MENUON_BIT              =     6
ICONSON_BIT             =     5



; ********************************************************************************************
;                                          Grafica
; ********************************************************************************************


;Costanti per le dimensioni dello schermo

.ifdef                  C-128
.if                     C-128
SC_BYTE_WIDTH           =     80           ;C-128 a 80 colonne
```

```
SC_PIX_WIDTH            =    640
.else
SC_BYTE_WIDTH           =    40         ;C-128 a 40 colonne
SC_PIX_WIDTH            =    320
.endif
.else
SC_BYTE_WIDTH           =    40         ;C-64 a 40 colonne
SC_PIX_WIDTH            =    320
.endif

SC_PIX_HEIGHT           =    200
SC_SIZE                 =    SC_PIX_WIDTH * SC_PIX_HEIGHT

;Costanti per controllare i bit di dispBufferOn
;dispBufferOn controlla quale schermo e' interessato dai comandi grafici

ST_WR_FORE              =    $80        ;Scrive sullo schermo principale
ST_WR_BACK              =    $40        ;Scrive sullo schermo nascosto
ST_WRGS_FORE            =    $20        ;La stringa grafica agisce solo
                                        ;sullo schermo principale

;Valori per la stringa grafica

MOVEPENTO               =    1          ;Muove il pennello alla coordinata x, y
LINETO                  =    2          ;Disegna una linea a x, y
RECTANGLETO             =    3          ;Disegna un rettangolo a x, y
;PENFILL                =    4          ;Riempie con la matrice grafica corrente,
                                        ;questo comando non e' ancora disponibile
                                        ;nel Kernel, e viene ignorato se presente
NEWPATTERN              =    5          ;Seleziona una nuova matrice grafica
ESC_PUTSTRING           =    6          ;Inizia l'interpretazione dei dati che
                                        ;seguono come per la routine i_PutString
FRAME_RECTO             =    7          ;Crea il bordo (cornice) a un rettangolo
PEN_X_DELTA             =    8          ;Muove il pennello orizzontalmente
                                        ;di una distanza specificata dalla word
                                        ;segnata che segue
PEN_Y_DELTA             =    9          ;Muove il pennello verticalmente
                                        ;di una distanza specificata dal byte
                                        ;segnato che segue
PEN_XY_DELTA            =    10         ;Muove il pennello delle distanze specificate
                                        ;dalla word e dal byte segnati che seguono
```

```
;I colori riconosciuti dal processore VIC

BLACK                    = 0
WHITE                    = 1
RED                      = 2
CYAN                     = 3
PURPLE                   = 4
GREEN                    = 5
BLUE                     = 6
YELLOW                   = 7
ORANGE                   = 8
BROWN                    = 9
LTRED                    = 10
DKGREY                   = 11
GREY                     = 12
MEDGREY                  = GREY
LTGREEN                  = 13
LTBLUE                   = 14
LTGREY                   = 15

;Valori per la routine PutDecimal

SET_LEFTJUST             =    %10000000   ;Giustificazione a sinistra
SET_RIGHTJUST            =    %00000000   ;Giustificazione a destra
SET_SUPRESS              =    %01000000   ;Tronca gli zeri
SET_NOSUPRESS            =    %00000000   ;Non tronca gli zeri


; *********************************************************************************************
;                                            Icone
; *********************************************************************************************


CLICK_COUNT              =    30          ;Numero di interruzioni da contare
                                          ;Questa costante viene utilizzata per
                                          ;inizializzare la variabile dblClickCount
                                          ;quando un'icona viene selezionata
                                          ;la prima volta. dblClickCount viene
                                          ;decrementata a ogni interrupt, e se non
                                          ;vale 0 quando l'icona viene nuovamente
                                          ;selezionata, il flag della doppia pressione
                                          ;(r0H) viene passato alla routine
```

```
                                        ;di servizio dell'icona con il valore TRUE
                                        ;Se dblClickCount vale 0 quando l'icona
                                        ;viene selezionata, il flag viene passato
                                        ;con il valore FALSE

;Le seguenti costanti contengono i valori per la variabile iconSetFlag. Questi valori
;indicano come viene segnalata all'utente la selezione di un'icona. Se e' impostato ST_FLASH,
;ST_INVERT non ottiene alcun effetto

ST_FLASH                =       $80     ;Bit per indicare che quando l'utente seleziona
                                        ;l'icona, questa deve invertirsi per un istante
ST_INVERT               =       $40     ;Bit per indicare che quando l'utente seleziona
                                        ;l'icona, questa deve invertirsi

;Offset all'interno della struttura di definizione delle icone

OFF_NM_ICNS             =       0       ;Offset al byte che specifica il numero di icone
                                        ;nella struttura
OFF_IC_XMOUSE           =       1       ;Offset alla word che specifica la coordinata x
                                        ;del mouse
OFF_IC_YMOUSE           =       3       ;Offset al byte che specifica la coordinata y
                                        ;del mouse

;Offset all'interno di un gruppo di dati di definizione di un'icona

OFF_PIC_ICON            =       0       ;Offset al puntatore ai dati grafici dell'icona
OFF_X_ICON_POS          =       2       ;Offset alla coordinata x dell'icona
OFF_Y_ICON_POS          =       3       ;Offset alla coordinata y dell'icona
OFF_WDTH_ICON           =       4       ;Offset alla larghezza dell'icona
OFF_HEIGHT_ICON         =       5       ;Offset all'altezza dell'icona
OFF_SRV_RT_ICON         =       6       ;Offset al puntatore alla routine di servizio
                                        ;dell'icona
OFF_NX_ICON             =       8       ;Offset all'icona successiva nella struttura
                                        ;di definizione delle icone


; ******************************************************************************************
;                                       Flag
; ******************************************************************************************


;Valori per pressFlag
```

```
KEYPRESS_BIT            =    7           ;Altra pressione
INPUT_BIT               =    6           ;Cambiamenti di stato nel dispositivo di input
MOUSE_BIT               =    5           ;Pressione del pulsante del mouse

SET_KEYPRESS            =    %10000000   ;Altra pressione
SET_INPUTCHG            =    %01000000   ;Cambiamenti di stato nel dispositivo di input
SET_MOUSE               =    %00100000   ;Pressione del pulsante del mouse

;Valori per faultData

OFFTOP_BIT              =    7           ;Sconfinamento in alto del mouse
OFFBOTTOM_BIT           =    6           ;Sconfinamento in basso del mouse
OFFLEFT_BIT             =    5           ;Sconfinamento a sinistra del mouse
OFFRIGHT_BIT            =    4           ;Sconfinamento a destra del mouse
OFFMENU_BIT             =    3           ;Sconfinamento del mouse dai bordi di un menu

SET_OFFTOP              =    %10000000   ;Sconfinamento in alto del mouse
SET_OFFBOTTOM           =    %01000000   ;Sconfinamento in basso del mouse
SET_OFFLEFT             =    %00100000   ;Sconfinamento a sinistra del mouse
SET_OFFRIGHT            =    %00010000   ;Sconfinamento a destra del mouse
SET_OFFMENU             =    %00001000   ;Sconfinamento del mouse dai bordi di un menu
ANY_FAULT = SET_OFFTOP|SET_OFFBOTTOM|SET_OFFLEFT|SET_OFFRIGHT|SET_OFFMENU


; ****************************************************************************************
;                                  Tipi dei file GEOS
; ****************************************************************************************


;Questi valori sono memorizzati nel File Entry dei file e indicano il tipo del file
;in ambiente GEOS

NOT_GEOS                =    0           ;File C-64 non compatibile con GEOS

;Le seguenti costanti sono tipi di file GEOS predisposti per mantenere la compatibilita'
;con i file C-64 precedenti. Questi file hanno semplicemente il File Header Block
;e il File Entry aggiornato. In genere i tipi BASIC e ASSEMBLY sono caricabili
;da deskTop e vengono eseguiti con il Basic attivato

BASIC                   =    1           ;Programma in Basic C-64 con un blocco File Header
                                         ;associato (tipo Commodore PRG). Questo tipo puo'
                                         ;essere utilizzato con programmi che normalmente
```

```
                                          ;verrebbero caricati ed eseguiti in questo modo:
                                          ;LOAD "FILE", 8
                                          ;RUN

ASSEMBLY              =    2              ;Programma in Assembly C-64 con un blocco File
                                          ;Header associato (tipo Commodore PRG). Questo
                                          ;tipo puo' essere utilizzato con programmi che
                                          ;normalmente verrebbero caricati ed eseguiti in
                                          ;questo modo:
                                          ;LOAD "FILE", 8, 1
                                          ;SYS (Indirizzo d'esecuzione)

DATA                  =    3              ;File di dati non eseguibile (PRG, SEQ, USR)
                                          ;con un blocco File Header associato

;Le costanti che seguono indicano i tipi GEOS per i file applicazioni e di sistema. Tutti
;i file identificati con uno dei tipi GEOS che seguono dovrebbero essere di tipo
;Commodore USR

SYSTEM                =    4              ;File di sistema GEOS
DESK_ACC              =    5              ;File accessori da scrivania GEOS
APPLICATION           =    6              ;File applicazioni GEOS
APPL_DATA             =    7              ;File dati per un'applicazione GEOS
FONT                  =    8              ;File fonte carattere GEOS
PRINTER               =    9              ;Driver di stampa GEOS
INPUT_DEVICE          =    10             ;Driver del dispositivo di input GEOS
DISK_DEVICE           =    11             ;Driver del disk drive GEOS
SYSTEM_BOOT           =    12             ;File GEOS di caricamento del sistema (GEOS,
                                          ;GEOS BOOT, GEOS KERNEL)
TEMPORARY             =    13             ;File temporaneo. deskTop cancella automatica-
                                          ;mente tutti i file di questo tipo presenti sul
                                          ;disco che sta aprendo. Quando l'applicazione
                                          ;deve creare un file temporaneo, deve
                                          ;utilizzare il tipo TEMPORARY e iniziare il
                                          ;nome del file con il carattere di controllo
                                          ;PLAINTEXT. Per esempio:
                                          ;swapName:
                                          ;.byte PLAINTEXT, "My swap file",0
                                          ;In questo modo ci si assicura che il file
                                          ;temporaneo non vada a sostituire un file
                                          ;su disco con lo stesso nome
```

```
AUTO_EXEC              =    14         ;Applicazione che dev'essere automaticamente
                                       ;caricata e mandata in esecuzione dopo
                                       ;l'installazione del sistema, ma prima che sia
                                       ;eseguita deskTop. Questo tipo viene riconosciuto
                                       ;solo dalla versione 1.3 del Kernel e successive
INPUT_128              =    15         ;Driver di input per il 128

.ifdef                 C-128
.if                    C-128
NUM_FILE_TYPES         =    16         ;Numero di tipi dei file ammessi da GEOS
                                       ;(compreso il tipo 0 NOT_GEOS)

.endif
.else
NUM_FILE_TYPES         =    15         ;Numero di tipi dei file ammessi da GEOS
                                       ;(compreso il tipo 0 NOT_GEOS)
.endif

                                       ;Tipi di struttura dei file gestiti da GEOS
                                       ;Ogni tipo di struttura identifica
                                       ;l'organizzazione dei blocchi del file
                                       ;su disco, e non ha attinenza con il formato
                                       ;dei dati nei blocchi

SEQUENTIAL             =    0          ;Struttura a blocchi T/S concatenati (come per
                                       ;esempio i file Commodore PRG e SEQ)
VLIR                   =    1          ;Struttura a record indicizzati di lunghezza
                                       ;variabile (utilizzata per fonti carattere,
                                       ;documenti ecc.). Questa struttura e' usata
                                       ;solo in ambiente GEOS


; ****************************************************************************************
;                              Tipi dei file Commodore
; ****************************************************************************************


;Sono i tipi dei file riconosciuti dal DOS del 1541

DEL                    =    0          ;File DELeted (cancellato)
SEQ                    =    1          ;File SEQuential (sequenziale)
PRG                    =    2          ;File PRoGram (programma)
```

```
USR                  =   3      ;File USeR (utente)
REL                  =   4      ;File RELative (relativo)
CBM                  =   5      ;File di protezione della BAM, attualmente
                                ;installato solo nel drive 1581. Viene impiegato
                                ;per proteggere particolari blocchi e tracce del
                                ;disco dall'esecuzione del comando Validate

TOTAL_BLOCKS         =   664    ;Numero di blocchi su disco, senza includere la
                                ;traccia della directory


; ****************************************************************************************
;                           Il Directory Header Block
; ****************************************************************************************

OFF_TO_BAM           =   4      ;Offset alla prima entry nella BAM
OFF_DISK_NAME        =   144    ;Offset alla stringa contenente il nome del disco
OFF_OP_TR_SC         =   171    ;Offset all'indirizzo T/S del blocco Off Page
                                ;contenente gli 8 File Entry dei file riposti
                                ;sulla scrivania
OFF_GS_ID            =   173    ;Posizione nel Directory Header Block della
                                ;stringa di identificazione GEOS
OFF_GS_DTYPE         =   189    ;Posizione dove si trova il tipo di disco nel
                                ;formato GEOS. Attualmente e' 0 per i dischi
                                ;normali, 'B' per i dischi di caricamento
                                ;(BOOT) e 'P' per i dischi Master. Questa
                                ;locazione e' azzerata sulle copie destinazione
                                ;di qualsiasi disco
OFF_GS_SERIAL        =   190    ;Posizione nella quale deskTop, e alcune
                                ;applicazioni GEOS, memorizzano il numero di serie
                                ;del Kernel. Questo accorgimento e' riservato
                                ;solo a dischi di tipo 'B' e 'P'. Si rivela
                                ;utile nel caso che l'utente rovini il proprio
                                ;disco GEOS, e noi dobbiamo spedirgliene
                                ;un altro in sostituzione. Il programma di
                                ;installazione legge il numero di serie
                                ;memorizzato sul master disk, e aggiorna
                                ;il numero di serie del Kernel per il disco
                                ;sostitutivo. Nota: Questa informazione e'
                                ;memorizzata in una word. Il numero seriale,
                                ;prima di essere memorizzato, viene ruotato a
                                ;sinistra di un bit. Sui dischi di copia,
                                ;questa locazione viene azzerata
```

```
; ***************************************************************************************
;                                       File Entry
; ***************************************************************************************

ST_WR_PR            =   $40     ;Bit di protezione del file, bit 6 del byte 0
                                ;nel File Entry. Questa costante e' utilizzabile
                                ;per verificare e/o impostare lo stato del bit
                                ;di protezione dei file Commodore

FRST_FILE_ENTRY     =   2       ;Offset al primo File Entry nel blocco
                                ;della directory

;Offset all'interno di un File Entry

OFF_CFILE_TYPE      =   0       ;Offset all'indicatore del tipo standard
                                ;Commodore del file
OFF_INDEX_PTR       =   1       ;Offset al puntatore T/S del blocco indice
                                ;(struttura VLIR)
OFF_DE_TR_SC        =   1       ;Offset al puntatore T/S del primo blocco
                                ;del file (struttura SEQ)
OFF_FNAME           =   3       ;Offset al nome del file
OFF_GHDR_PTR        =   19      ;Offset all'indirizzo T/S del blocco File Header
                                ;del file
OFF_GSTRUC_TYPE     =   21      ;Offset all'identificatore della struttura GEOS
                                ;del file
OFF_GFILE_TYPE      =   22      ;Offset all'indicatore del tipo GEOS del file
OFF_YEAR            =   23      ;Offset al byte anno (primo byte della data)
OFF_SIZE            =   28      ;Offset alla dimensione del file in blocchi
OFF_NXT_FILE        =   32      ;Offset al File Entry seguente nel blocco della
                                ;directory


; ***************************************************************************************
;                                  Il blocco File Header
; ***************************************************************************************


;Offset all'interno del blocco File Header di un file

O_GHIC_WIDTH        =   2       ;Byte che indica la larghezza in byte
                                ;dell'icona associata al file
O_GHIC_HEIGHT       =   3       ;Byte che indica l'altezza in byte dell'icona
```

```
O_GHIC_PIC          =   4      ;64 byte per il disegno dell'icona
O_GHCMDR_TYPE       =   68     ;Tipo Commodore del file per il File Entry
O_GHGEOS_TYPE       =   69     ;Tipo GEOS del file per il File Entry
O_GHSTR_TYPE        =   70     ;Struttura GEOS del file
O_GHST_ADDR         =   71     ;Word, indirizzo di caricamento del file
                               ;(da dove e' stato salvato)
O_GHEND_ADDR        =   73     ;Word, indirizzo di fine file in memoria
O_GHST_VEC          =   75     ;Word, indirizzo di esecuzione se il file
                               ;e' un'applicazione
O_GHFNAME           =   77     ;20 byte, nome permanente
O_128_FLAGS         =   96     ;Byte contenente alcuni flag che indicano se
                               ;l'applicazione e' in grado di funzionare in
                               ;ambiente GEOS 128 a 40 colonne e a 80 colonne
                               ;Questi flag interessano i file di tipo
                               ;APPLICATION, DESK_ACC, e AUTO_EXEC
                               ;Bit Significato
                               ;7   0 se funziona in modo a 40 colonne
                               ;6   1 se funziona in modo a 80 colonne
O_GHFONTID          =   130    ;Se il file e' una fonte carattere, offset
                               ;ai 10 bit contenenti il numero Font ID della
                               ;fonte. Questa informazione identifica anche
                               ;il primo corpo carattere della fonte
O_GHPOINT_SIZE      =   132    ;Se il file e' una fonte carattere, offset alla
                               ;lista dei corpi carattere disponibili (massimo
                               ;15 corpi)
O_GHSET_LENGHTS     =   97     ;Se il file e' una fonte carattere, offset alle
                               ;dimensioni, in byte, di ogni set all'interno
                               ;della fonte. I set disponibili sono listati a
                               ;O_GHPOINT_SIZE. Ogni dimensione e' composta
                               ;da due byte
O_GHP_DISK          =   97     ;20 byte, nome del disco sul quale si trova
                               ;l'applicazione parente (solo se il file e'
                               ;un file dati). Gli ultimi 4 caratteri sono
                               ;nulli, mentre i caratteri 12-15 sono
                               ;la versione (per esempio, 1.3)
O_GHP_FNAME         =   117    ;20 byte, nome dell'applicazione parente
                               ;(solo se il file e' un file dati)
O_GH_AUTHOR         =   97     ;Offset al nome dell'autore. Occupa lo stesso
                               ;spazio nel blocco utilizzato per il nome
                               ;del disco parente in quanto le due informazioni
                               ;non possono essere presenti contemporaneamente
                               ;nel file
```

```
O_GHINFO_TXT            =   $A0      ;Offset allo spazio nel blocco allocato per
                                     ;contenere il testo associato al file che
                                     ;compare nel box get info di deskTop

; ****************************************************************************************
;                        Costanti per la routine GetFile
; ****************************************************************************************

;Le costanti che seguono definiscono le opzioni di caricamento per la routine GetFile e, in
;generale, per tutte le routine di caricamento da disco (Ldxxxx). Queste costanti si usano
;per impostare lo stato dei bit contenuti nel registro loadOpt

ST_LD_AT_ADDR           =   $01      ;Carica il file all'indirizzo specificato e non
                                     ;a quello originario
ST_LD_DATA              =   $80      ;(Solo per le applicazioni) il caricamento e'
                                     ;avvenuto tramite la selezione di un file dati
                                     ;L'applicazione deve caricare ed elaborare
                                     ;il file dati selezionato, il cui nome e'
                                     ;puntato da r3
ST_PR_DATA              =   $40      ;(Solo per le applicazioni) il caricamento
                                     ;dell'applicazione e' avvenuto per stampare un
                                     ;file dati. L'applicazione riceve in r3
                                     ;il puntatore al nome del file da stampare


; ****************************************************************************************
;                                    Disco
; ****************************************************************************************

;Costanti per le routine d'accesso al disco di basso livello

N_TRACKS                =   35       ;Numero di tracce disponibili sui dischi
                                     ;per il 1541
DIR_TRACK               =   18       ;Traccia sul disco riservata per la directory
DIR_1581_TRACK          =   40       ;Traccia sul disco riservata per la directory
                                     ;in un disk drive 1581
                                     ;Le costanti per la variabile "driveType"
                                     ;i due bit piu' significativi in driveType
                                     ;hanno particolari significati (non possono
                                     ;essere impostati a 1 contemporaneamente):
                                     ;Bit 7: se impostato a 1, disco virtuale su RAM
                                     ;Bit 6: se impostato a 1, shadowed disk
```

```
DRV_NULL            =   0       ;Non e' presente alcun drive all'indirizzo
                                ;di dispositivo corrente
DRV_1541            =   1       ;Disk drive Commodore 1541
DRV_1571            =   2       ;Disk drive Commodore 1571
DRV_1581            =   3       ;Disk drive Commodore 1581

DRV_NETWORK         =   15      ;Drive per GEOS geoNet "drive"

DK_NM_ID_LEN        =   18      ;Numero di caratteri che compongono il nome
                                ;del disco

;Comandi d'accesso al disco

MAX_CMND_STR        =   32      ;Lunghezza massima di una stringa comandi
                                ;per il drive
DIR_ACC_CHAN        =   13      ;Canale di default per l'accesso diretto
REL_FILE_NUM        =   9       ;Numero logico e canale per i file relativi
CMND_FILE_NUM       =   15      ;Numero logico e canale per i file comandi

;Le due costanti che seguono individuano, all'interno di un buffer contenente
;una stringa comando, i numeri di traccia e settore per l'accesso diretto al disco

TRACK               =   9       ;Offset al byte basso contenente il codice
                                ;ASCII del numero decimale della traccia
SECTOR              =   12      ;Offset al byte basso contenente il codice
                                ;ASCII del numero decimale del settore

; ***************************************************************************************
;                                   Errori da disco
; ***************************************************************************************

;Le costanti che seguono individuano i codici d'errore restituiti dalle routine che
;effettuano accessi diretti al drive

NOT_BLOCKS          =   1       ;Questo errore viene restituito quando
                                ;il numero di blocchi da trasferire non
                                ;corrisponde a quello incontrato dalle routine
                                ;GetBlocks o PutBlocks. Per esempio, questo
                                ;codice viene restituito da GetBlocks se
                                ;incontra l'ultimo blocco del file prima che
                                ;sia stato trasferito il numero di blocchi
                                ;richiesto
```

```
INV_TRACK           =   2   ;Questo errore si verifica quando si tenta di
                            ;accedere, in scrittura o in lettura,
                            ;al settore di una traccia inesistente
                            ;(traccia 0 o traccia > N_TRACKS)
INSUFF_SPACE        =   3   ;E' restituito da BlockAlloc o SaveFile se il disco
                            ;non contiene il numero di settori allocabili
                            ;necessario
FULL_DIRECTORY      =   4   ;Questo errore viene restituito da GetFreeDirBlk
                            ;se dalla pagina della directory fino all'ultimo
                            ;blocco non e' presente neanche un File Entry
                            ;libero. Nota: se non vengono utilizzati tutti i 18
                            ;blocchi possibili, GetFreeDirBlk aggiunge una
                            ;pagina alla directory, ma se tutti i blocchi sono
                            ;gia' stati allocati viene restituito il codice di
                            ;errore per indicare che dalla pagina indicata
                            ;fino al termine della directory non sono presenti
                            ;blocchi liberi. La presenza di questo errore non
                            ;indica necessariamente che la directory non
                            ;contiene neanche un File Entry libero, ma solo
                            ;che non sono presenti File Entry liberi
                            ;dalla pagina indicata fino al termine della
                            ;directory. Se la pagina da cui parte la ricerca
                            ;e' la numero 0, questo codice d'errore indica che
                            ;la directory e' completamente piena e non puo'
                            ;contenere altri File Entry (in questo caso i file
                            ;presenti sul disco sono 144)
FILE_NOT_FOUND      =   5   ;Viene restituito quando, scorrendo la directory,
                            ;il file richiesto non viene incontrato
BAD_BAM             =   6   ;Indica che la BAM memorizzata nel buffer
                            ;curDirHead presenta un errore:
                            ;1) il numero di blocchi liberi non corrisponde a
                            ;quello ottenuto calcolando i bit della BAM
                            ;2) durante la cancellazione di un file, un blocco
                            ;del file presenta gia' nella BAM lo stato di
                            ;disallocato (libero)
UNOPENED_VLIR       =   7   ;Questo errore si verifica quando si tenta
                            ;l'accesso a un record di un file VLIR prima che
                            ;questo sia stato aperto con la routine
                            ;OpenRecordFile
INV_RECORD          =   8   ;Questo errore si verifica quando si tenta
                            ;l'accesso a un record del file VLIR che non esiste
                            ;Questo errore non e' fatale, e puo' essere
```

```
                                         ;utilizzato per muovere il puntatore ai record
                                         ;lungo la tavola indice in modo che arrivi al
                                         ;termine della lista
OUT_OF_RECORDS              =     9      ;Questo errore si verifica quando si tenta di
                                         ;inserire/appendere un ulteriore record a un file
                                         ;che gia' contiene il massimo numero di record
                                         ;possibile (attualmente 127)
STRUCT_MISMAT               =     10     ;Questo errore si verifica quando una routine
                                         ;realizzata per gestire una particolare struttu-
                                         ;ra di file viene impiegata per agire su file con
                                         ;struttura diversa
BFR_OVERFLOW                =     11     ;Questo errore si verifica durante la chiamata a
                                         ;ReadFile, quando nel corso del caricamento il
                                         ;numero di byte del file oltrepassa il massimo
                                         ;specificato all'atto della chiamata alla
                                         ;routine
CANCEL_ERR                  =     12     ;Questa chiamata d'errore e' stata realizzata per
                                         ;permettere al programmatore di impiegare un box
                                         ;di dialogo durante accessi al disco di basso
                                         ;livello, e di utilizzare l'icona CANCEL per
                                         ;bloccare inmaniera distruttiva la procedura.Per
                                         ;esempio, se l'applicazione osserva che e' stata
                                         ;selezionata l'icona CANCEL, deve caricare in x il
                                         ;codice dell'errore CANCEL_ERR prima di restitui-
                                         ;re il controllo (la convenzione per gli errori da
                                         ;disco e' che siano restituiti in x prima del
                                         ;ritorno). La routine di livello superiore e'
                                         ;cosi' in grado di determinare che il codice
                                         ;d'errore restituito non corrisponde in realta' a
                                         ;un errore avvenuto durante l'accesso al disco, ma
                                         ;alla richiesta dell'utente di sospendere
                                         ;l'operazione
DEV_NOT_FOUND               =     13     ;Questo errore si verifica quando la routine
                                         ;"Listen" del Kernel del C-64 viene impiegata
                                         ;per aprire un canale di comunicazione con un
                                         ;dispositivo sul bus seriale, e il dispositivo non
                                         ;e' attivo
INCOMPATIBLE                =     14     ;Questo errore si verifica quando vengono
                                         ;richiesti il caricamento e l'esecuzione di un
                                         ;programma che non puo' essere eseguito con il
                                         ;modo grafico correntemente attivato da GEOS 128
```

```
HDR_NOT_THERE           =    $20    ;Questo errore si verifica quando il blocco File
                                    ;Header di un file non e' presente
NO_SYNC                 =    $21    ;Questo errore si verifica quando il drive non
                                    ;riesce a trovare sul disco il marcatore di
                                    ;sincronismo. In genere la causa e' una di queste:
                                    ;il disco non e' inserito nel drive, il meccanismo
                                    ;di chiusura e' aperto, il disco non e' formattato
DBLK_NOT_THERE          =    $22    ;Il blocco di dati non e' presente nel settore
                                    ;richiesto
DAT_CHKSUM_ERR          =    $23    ;Il blocco su disco non verifica il controllo di
                                    ;somma (checksum). In questo il caso il blocco e'
                                    ;stato memorizzato male
WR_VER_ERR              =    $25    ;La verifica dei dati appena trasferiti in un
                                    ;settore del disco non ha successo. Normalmente
                                    ;accade se il blocco e' alterato
WR_PR_ON                =    $26    ;Questo errore si verifica quando si tenta di
                                    ;scrivere su un disco munito di fascetta
                                    ;protettiva
HDR_CHKSUM_ERR          =    $27    ;Il blocco header non verifica il controllo di
                                    ;somma (checksum)
DSK_ID_MISMAT           =    $29    ;La ID letta da disco non corrisponde alla ID
                                    ;interna prevista. Normalmente si verifica questo
                                    ;errore se: viene cambiato il disco senza che venga
                                    ;inoltrato il comando NewDisk, l'utente commette
                                    ;un errore nella manipolazione dei dischi da
                                    ;inserire
BYTE_DEC_ERR            =    $2E    ;Errore nella codificazione del flusso di dati
                                    ;proveniente dal disco
DOS_MISMATCH            =    $73    ;L'identificatore del DOS del drive non
                                    ;corrisponde


; ********************************************************************************************
;                                        Box di dialogo
; ********************************************************************************************


DEF_DB_POS              =    $80    ;Flag per indicare la posizione e le dimensioni
                                    ;del BD di default (gestito quindi dal Kernel).
SET_DB_POS              =    $00    ;Flag per indicare che l'applicazione deve prov-
                                    ;vedere a gestire le dimensioni del BD e la sua
                                    ;posizione.
```

```
;Tavola dei comandi per i box di dialogo (BD)

;Le icone di sistema disponibili per i BD. Queste icone sono predefinite e facilitano
;la realizzazione dei BD. Nella maggior parte dei casi causano la chiusura del BD e la
;restituzione del controllo all'applicazione. Il numero dell'icona selezionata viene
;passato all'applicazione tramite la variabile sysDBData
;Nota: in tutti i comandi per i BD, le posizioni specificate (tramite le coordinate x e y)
;devono riferirsi all'angolo superiore sinistro del BD, e non dello schermo.
;Normalmente questi offset sono in pixel, ma in alcuni casi la coordinata x puo' essere
;richiesta in byte (per esempio nel caso delle icone)

;Per tutti i comandi delle icone di sistema, i due byte che seguono indicano
;rispettivamente la coordinata x in byte e la coordinata y in pixel

OK                      =   1       ;Icona di sistema per i BD: "OK"
CANCEL                  =   2       ;Icona di sistema per i BD: "CANCEL"
YES                     =   3       ;Icona di sistema per i BD: "YES"
NO                      =   4       ;Icona di sistema per i BD: "NO"
OPEN                    =   5       ;Icona di sistema per i BD: "OPEN"
DISK                    =   6       ;Icona di sistema per i BD: "DISK"
FUTURE1                 =   7       ;Riservata per future icone di sistema
FUTURE2                 =   8       ;Riservata per future icone di sistema
FUTURE3                 =   9       ;Riservata per future icone di sistema
FUTURE4                 =   10      ;Riservata per future icone di sistema

;Altri comandi per i box di dialogo

DBTXTSTR                =   11      ;Comando per visualizzare un stringa ASCII. Il
                                    ;comando dev'essere seguito dall'indirizzo della
                                    ;stringa a terminazione nulla e dalla posizione
                                    ;all'interno del BD
DBVARSTR                =   12      ;Comando per la visulizzazione di stringhe
                                    ;diverse. Dev'essere seguito dall'indirizzo di
                                    ;un registro in pagina 0. Questo registro puo' di
                                    ;volta in volta contenere l'indirizzo di una
                                    ;diversa stringa da visualizzare. Segue
                                    ;l'indicazione della posizione all'interno
                                    ;del BD
DBGETSTRING             =   13      ;Accetta una stringa ASCII dall'utente e la
                                    ;memorizza in un buffer. Il comando e' seguito
                                    ;dalla posizione dell'eco della stringa sullo
                                    ;schermo, dall'indirizzo del registro che
```

```
                                ;contiene il puntatore al buffer da utilizzare e
                                ;dal massimo numero di caratteri accettabili
DBSYSOPV        =       14      ;Il comando non ha parametri. Qualsiasi pressione
                                ;del pulsante del mouse su una zona dello schermo
                                ;che non e' un'icona, causa la chiusura del BD
                                ;e il ritorno all'applicazione
DBGRPHSTR       =       15      ;Questo comando manda in esecuzione una stringa
                                ;grafica. Il comando dev'essere seguito da un
                                ;puntatore alla stringa grafica in memoria.
DBGETFILES      =       16      ;Questo comando richiede alla struttura di gestio-
                                ;ne dei BD di visualizzare, nella parte sinistra
                                ;della finestra, un box con la lista di tutti i file
                                ;del tipo specificato presenti sul disco. Se il
                                ;disco contiene un numero di file del tipo indicato
                                ;maggiore di quello che trova posto nel box,
                                ;appaiono nella parte inferiore due frecce che
                                ;possono comandare lo scroll verticale della
                                ;lista. Il comando dev'essere seguito dalla
                                ;posizione dell'angolo superiore del box. La rou-
                                ;tine di gestione del comando si aspetta che in
                                ;r7L sia memorizzato il puntatore al tipo di file,
                                ;in r5 il puntatore al buffer entro il quale
                                ;memorizzare la lista dei nomi di file, e r10L
                                ;contenga il puntatore al nome permanente
                                ;utilizzato come seconda chiave di ricerca. Se
                                ;questa ulteriore specificazione non e'
                                ;richiesta, r10L dev'essere azzerato
DBOPVEC         =       17      ;L'indirizzo che segue il comando indica il
                                ;vettore otherPressVec impostato
                                ;dall'applicazione. Impartendo questo comando,
                                ;se l'utente preme il pulsante del mouse in
                                ;una zona che non sia un'icona o un menu,
                                ;GEOS esegue la routine puntata dal vettore
                                ;otherPressVec
DBUSRICON       =       18      ;Il comando permette all'applicazione di definire
                                ;un'icona non standard da inserire nella struttu-
                                ;ra delle icone del BD. La word che segue il
                                ;comando individua la tavola di definizione
                                ;dell'icona. Segue la posizione dell'icona
                                ;all'interno del BD
DB_USR_ROUT     =       19      ;Questo comando definisce una routine, creata
                                ;dall'applicazione, che GEOS esegue dopo aver
```

```
                                        ;visualizzato il BD. L'indirizzo della routine
                                        ;viene subito dopo il comando

;Le seguenti costanti sono gli offset da utilizzare all'interno delle tavola di
;definizione dei BD per indicare i parametri significativi

OFF_DB_FORM          =    0             ;Offset all'indicatore della forma (per
                                        ;esempio, con ombra o meno).
OFF_DB_TOP           =    1             ;Offset alla posizione del lato superiore del BD
OFF_DB_BOT           =    2             ;Offset alla posizione del lato inferiore del BD
OFF_DB_LEFT          =    3             ;Offset alla posizione del lato sinistro del BD
OFF_DB_RIGHT         =    5             ;Offset alla posizione del lato destro del BD
OFF_DB_1STCMD        =    7             ;Offset al primo comando della tavola di
                                        ;definizione del BD quando e' l'applicazione a
                                        ;definirne le dimensioni e la posizione.
                                        ;La costante che segue stabilisce quante icone
                                        ;possono essere definite all'interno della
                                        ;tavola di definizione. Questa costante viene
                                        ;utilizzata per allocare lo spazio in RAM per
                                        ;le icone del BD

MAX_DB_ICONS         =    8             ;Numero massimo di icone in un BD

;La costante che segue viene utilizzata per specificare gli offset x e y dal BD per
;visualizzare l'ombra, se e' richiesta. Questa costante dev'essere un multiplo di 8, per
;evitare una non compatibilita' con future versioni a colori.

DB_SHAD_OFFSET       =    8             ;Offset dai bordi del BD per visualizzare l'ombra

;Le costanti che seguono individuano le dimensioni delle icone standard per i BD messe
;a disposizione da GEOS.

SYSDBI_WIDTH         =    6             ;Larghezza in byte dell'icona
SYSDBI_HEIGHT        =    16            ;Altezza in pixel dell'icona

;Le seguenti costanti sono utilizzate per gestire il comando BDGETFILES

DB_F_BOX_WIDTH       =    124           ;Larghezza in pixel del box nel quale appaiono
                                        ;i nomi dei file del tipo specificato
DB_F_BOX_HEIGHT      =    88            ;Altezza in pixel del box nel quale appaiono
                                        ;i nomi dei file del tipo specificato
```

```
DB_F_SCR_ICN_X_OFF      =   7     ;Offset in byte dalla linea verticale
                                  ;individuata dal lato sinistro del box
                                  ;dei file, alle icone per lo scroll
DB_F_SCR_ICN_HEIGHT     =   14    ;Altezza dell'icona di scroll dal lato
                                  ;inferiore del box
DBF_LINE_HEIGHT         =   16    ;Altezza, dal lato inferiore del box, che
                                  ;individua la posizione della linea di separazio-
                                  ;ne fra l'area delle icone e quella per i nomi
                                  ;dei file
MAX_DB_FILES            =   15    ;Numero massimo di file i cui nomi possono
                                  ;essere inseriti nel buffer
DB_FSCR_ICN_WIDTH       =   3     ;Larghezza, in byte, dell'icona utilizzata
                                  ;per lo scroll dei nomi
DB_FSCR_ICN_HEIGHT      =   12    ;Altezza, in pixel, dell'icona utilizzata per
                                  ;lo scroll dei nomi
DBF_Y_LINE_OFF          =   14    ;Offset y fra i diversi nomi dei file
DB_TXT_BASE_OFF         =   9     ;Offset, dal lato superiore dello spazio
                                  ;rettangolo, della posizione della linea di base
                                  ;per il testo che individua il nome del file

;Altre costanti per i box di dialogo

;Queste costanti definiscono le dimensioni e la posizione del box di dialogo standard,
;e altre posizioni standard per visualizzare testi e icone all'interno di un BD

DEF_DB_TOP              =   32    ;Coordinata y del lato superiore del BD di default
DEF_DB_BOT              =   127   ;Coordinata y del lato inferiore del BD di default
DEF_DB_LEFT             =   64    ;Coordinata x del lato sinistro del BD di default
DEF_DB_RIGHT            =   255   ;Coordinata x del lato destro del BD di default

TXT_LN_X                =   16    ;Coordinata x standard per i testi
TXT_LN_1_Y              =   16    ;Offset verticali standard per le linee di testo
TXT_LN_2_Y              =   32
TXT_LN_3_Y              =   48
TXT_LN_4_Y              =   64
TXT_LN_5_Y              =   80

DBI_X_0                 =   1     ;Offset in byte alla coordinata x dell'icona
                                  ;standard di sinistra
DBI_X_1                 =   9     ;Offset in byte alla coordinata x dell'icona
                                  ;standard di centro
DBI_X_2                 =   17    ;Offset in byte alla coordinata x dell'icona
                                  ;standard di destra
```

```
DBI_Y_0             =   8           ;Offset in pixel alla coordinata y dell'icona
                                    ;standard in alto
DBI_Y_1             =   40          ;Offset in pixel alla coordinata y dell'icona
                                    ;standard in centro
DBI_Y_2             =   72          ;Offset in pixel alla coordinata y dell'icona
                                    ;standard in basso

; ******************************************************************************************
;                                   Microprocessore VIC
; ******************************************************************************************


GRBANK0             =   %11         ;Indica che il banco da 16K visto dal VIC e' il
                                    ;primo: $0000 - $3FFF
GRBANK1             =   %10         ;Indica che il banco da 16K visto dal VIC e' il
                                    ;secondo: $4000 - $7FFF
GRBANK2             =   %01         ;Indica che il banco da 16K visto dal VIC e' il
                                    ;terzo: $8000 - $BFFF
GRBANK3             =   %00         ;Indica che il banco da 16K visto dal VIC e' il
                                    ;quarto: $C000 - $FFFF

SKIPFLAG            =   $AA         ;Flag per indicare che il particolare dato
                                    ;all'interno della tavola di'inizializzazione
                                    ;del VIC dev'essere ignorato

MOUSE_SPRNUM        =   0           ;Numero dello sprite utilizzato per il mouse

VIC_YPOS_OFF        =   50          ;Offset verticale per collocare uno sprite
                                    ;hardware in alto sullo schermo. Questo valore
                                    ;viene utilizzato per convertire le coordinate
                                    ;GEOS degli sprite in coordinate hardware
VIC_XPOS_OFF        =   24          ;Offset orizzontale per collocare uno sprite
                                    ;hardware in alto sullo schermo. Questo valore
                                    ;viene utilizzato per convertire le coordinate
                                    ;GEOS degli sprite in coordinate hardware

ALAMMASK            =   %00000100   ;Maschera per il bit d'allarme nel registro di
                                    ;controllo degli interrupt per il VIC
```

```
; ************************************************************************************************
;              Bit di salvataggio dello schermo per gli accessori da scrivania
; ************************************************************************************************

FG_SAVE                     =     %10000000   ;Salva e ripristina la parte di schermo
                                              ;interessata
CLR_SAVE                    =     %01000000   ;Salva e ripristina le informazioni del colore
```

B APPENDICE B: VARIABILI GEOS GLOBALI

```
;***********************************************************************************************
;
;                                          geosMemoryMap
;
;Questo file contiene una descrizione della mappa di memoria di GEOS e delle allocazioni
;in pagina 0 e in RAM sistema delle variabili utilizzate dal sistema operativo
;In questo file sono richiamate molte costanti definite nell'appendice A
;
;
;
;
;***********************************************************************************************
```

Mappa di memoria di GEOS

Numero di byte Decimale	Indirizzi del range Esadecimale	Descrizione
1	0000	Registro direzione dati del 6510
1	0001	Registro di I/O del 6510
110	0002 - 006F	RAM in pagina 0 per GEOS e per le applicazioni
16	0070 - 007F	RAM in pagina 0 solo per le applicazioni, registri a2-a9
11	0080 - 008A	RAM in pagina 0 per Kernel C-64 e Basic
5	008B - 008F	RAM in pagina 0 per i disk driver di GEOS
107	0090 - 00FA	RAM in pagina 0 utilizzata dal Kernel C-64 e dal Basic
4	00FB - 00FE	RAM in pagina 0 solo per le applicazioni, registri a0-a1
1	00FF	Utilizzata dal Kernel del C-64 e dal Basic
256	0100 - 01FF	Stack del 6510
89	0200 - 0258	RAM a disposizione delle applicazioni per le variabili temporanee
219	0259 - 0333	RAM utilizzata dal Kernel del C-64
204	0334 - 03FF	RAM per le applicazioni, spazio programma
23552	0400 - 5FFF	RAM per i codici delle applicazioni
8000	6000 - 7F3F	RAM dello schermo nascosto
192	7F40 - 7FFF	RAM per le applicazioni
2560	8000 - 89FF	Buffer e variabili per GEOS
512	8A00 - 8BFF	Dati dei disegni degli sprite
1000	8C00 - 8FE7	Matrice video dei colori
16	8FE8 - 8FF7	RAM per GEOS
8	8FF8 - 8FFF	Puntatori agli sprite
4096	9000 - 9FFF	Codici di GEOS
8000	A000 - BF3F	Schermo principale o ROM del Basic
192	BF40 - BFFF	Tavole di GEOS o ROM del Basic
4096	C000 - CFFF	4K Kernel di GEOS, sempre residente
3584	D000 - DDFF	4K Kernel di GEOS o spazio di I/O
512	DE00 - DFFF	Nessuna RAM/ROM, spazio per REU
7808	E000 - FE7F	8K Kernel di GEOS o Kernel C-64
378	FE80 - FFF9	Driver di input o Kernel del C-64
6	FFFA - FFFF	Vettori di NMI, IRQ e Reset del 6510

```
;Locazioni di memoria del C-64

CPU_DDR         =           $0000   ;Registro direzione dati del 6510
CPU_DATA        =           $0001   ;Registro di I/O del 6510

STATUS          =           $0090   ;Registro di stato del C-64
curDevice       =           $00BA   ;Dispositivo seriale corrente
irqvec          =           $0314   ;Vettore IRQ (word)
brkvec          =           $0316   ;Vettore BRK (word)
nmivec          =           $0318   ;Vettore NMI (word)
kernelVectors   =           $031A   ;Locazione dei vettori del Kernel
vicbase         =           $D000   ;Indirizzo base dell'interfaccia video
sidbase         =           $D400   ;Indirizzo base dell'interfaccia musicale
ctab            =           $D800   ;Mappa dei colori
cia1base        =           $DC00   ;Prima interfaccia di comunicazione (CIA)
cia2base        =           $DD00   ;Seconda interfaccia di comunicazione (CIA)
EXP_BASE        =           $DF00   ;Indirizzo base della porta di comunicazione con
                                    ;l'espansione RAM
kernelList      =           $FD36   ;Lista di vettori per il Kernel C-64
.ifdef          C-128
.if             C-128
mmu             =           $D500
VDC             =           $D600
.endif
.endif

;La variabile che segue viene di solito definita solo se C-128 = 1. E' utile quando
;un programma per il C-64 deve interagire con il Kernel del C-128

config          =           $FF00   ;Indirizzo del registro di configurazione
                                    ;nel C-128

;Indirizzi delle routine del Kernel del C-64. Per maggiori informazioni sul loro impiego,
;consultare il manuale pubblicato dalla Commodore Italiana, "Commodore 64,
;Guida di riferimento per il programmatore"

Acptr           =           $FFA5   ;Accetta un carattere dal bus seriale
Chkin           =           $FFC6   ;Apre un canale di input associato al file logico
Chkout          =           $FFC9   ;Apre un canale di output associato al file logico
Chrin           =           $FFCF   ;Preleva un byte dal canale seriale
Chrout          =           $FFD2   ;Trasmette un byte attraverso il canale seriale
Ciout           =           $FFA8   ;Trasmette un byte sul bus seriale
Cint            =           $FF81   ;Inizializza lo schermo (viene utilizzata per
                                    ;restituire il controllo al Basic)
```

```
Clall          =    $FFE7       ;Chiude tutti i file aperti
Close          =    $FFC3       ;Chiude il file logico passato nell'accumulatore
ClrChn         =    $FFCC       ;Cancella i canali di I/O
Getin          =    $FFE4       ;Preleva un carattere dalla scansione della tastiera
                                ;(non viene usata)
Iobase         =    $FFF3       ;Restituisce l'indirizzo dello spazio di I/O in
                                ;memoria (non viene usata)
Ioinit         =    $FF84       ;Inizializza tutti i dispositivi di I/O
Listen         =    $FFB1       ;Ordina a un dispositivo di mettersi in ascolto sul bus
                                ;seriale
Load           =    $FFD5       ;Carica un file da un dispositivo di input e lo
                                ;memorizza
Membot         =    $FF9C       ;Imposta o legge l'inizio della memoria Basic
Memtop         =    $FF99       ;Imposta o legge la fine della memoria Basic
Open           =    $FFC0       ;Apre un file logico
Restor         =    $FF8A       ;Ripristina i vettori di sistema e di interrupt
                                ;copiandoli dalla tavola a $FD30 e memorizzandoli a
                                ;$0314
Save           =    $FFD8       ;Salva una particolare area di memoria sul
                                ;dispositivo di output
Scnkey         =    $FF9F       ;Routine di scansione della tastiera
Second         =    $FF93       ;Trasmette l'indirizzo secondario al dispositivo in
                                ;ascolto
Setlfs         =    $FFBA       ;Prepara i parametri di un file logico
Setmsg         =    $FF90       ;Gestisce i messaggi del Kernel
Setnam         =    $FFBD       ;Imposta il nome del file
Tksa           =    $FF96       ;Trasmette un indirizzo secondario al dispositivo in
                                ;trasmissione
Unlsn          =    $FFAE       ;Trasmette sul bus seriale il comando di cessare
                                ;l'ascolto
Untlk          =    $FFAB       ;Trasmette sul bus seriale il comando di cessare la
                                ;trasmissione

;Definizioni degli spazi di memoria utilizzati da GEOS

SYSTEM_RAM     =    $0400       ;Inizio della RAM di sistema
APP_RAM        =    $0400       ;Inizio dello spazio per le applicazioni
BACK_SCR_BASE  =    $6000       ;Inizio dello schermo nascosto
PRINTBASE      =    $7900       ;Indirizzo al quale devono allocarsi i driver di
                                ;stampa
APP_VAR        =    $7F40       ;Spazio per le variabili delle applicazioni
OS_VARS        =    $8000       ;Inizio dello spazio per le variabili di sistema
```

```
SPRITE_PICS      =  $8A00        ;Inizio dei disegni degli sprite
COLOR_MATRIX     =  $8C00        ;Matrice video dei colori
DISK_BASE        =  $9000        ;Indirizzo base per i disk driver
SCREEN_BASE      =  $A000        ;Inizio dello schermo principale
OS_ROM           =  $C000        ;Inizio dei codici del sistema operativo
OS_JUMPTAB       =  $C100        ;Inizio della tavola contenente gli entry point per
                                 ;le routine di GEOS
MOUSE_JMP        =  $FE80        ;Inizio della jump table del driver di input
.ifndef         C-128
MOUSE_BASE       =  $FE80        ;Inizio in memoria del driver di input
END_MOUSE        =  $FFFA
.elif           C-128
MOUSE_BASE       =  $FE80        ;Inizio in memoria del driver di input
END_MOUSE        =  $FFFA
.else
MOUSE_BASE       =  $FD00        ;Inizio in memoria del driver di input
END_MOUSE        =  $FE80
.endif

;Locazioni a pagina $C000 che caratterizzano il Kernel correntemente installato

bootName         =  $C006        ;Inizio della stringa "GEOS BOOT"
version          =  $C00F        ;Byte che indica la versione di GEOS
nationality      =  $C010        ;Byte che indica la nazionalita' della versione
sysFlgCopy       =  $C012        ;Copia della variabile sysRAMFlg che viene salvata
                                 ;quando il controllo passa al Basic
dateCopy         =  $C018        ;Data della copia in anno, mese, giorno

;Entry point (punti d'ingresso) all'interno dei disk driver
;Questi indirizzi sono validi solo per disk drive diversi dal 1541 con GEOS 64, e per il driver
;del 1541 disponibile in ambiente GEOS 128:

Get1stDirEntry   =  $9030        ;Restituisce il primo File Entry in directory
GetNxtDirEntry   =  $9033        ;Restituisce il successivo File Entry in directory
AllocateBlock    =  $9048        ;Alloca un particolare blocco
ReadLink         =  $904B        ;E' simile a ReadBlock, ma restituisce solo i primi due
                                 ;byte del blocco

.ifdef          C-128
.if             C-128
jmpIndx          =  $9D80        ;Indirizzo di una routine necessaria al Kernel del
                                 ;C-128
```

```
.endif
.endif

;Indirizzi degli entry point all'interno del driver di stampa

InitForPrint    =   PRINTBASE       ;$7900 indirizzo di InitForPrint
StartPrint      =   PRINTBASE+3     ;$7903 indirizzo di StartPrint
PrintBuffer     =   PRINTBASE+6     ;$7906 indirizzo di PrintBuffer
StopPrint       =   PRINTBASE+9     ;$7909 indirizzo di StopPrint
GetDimensions   =   PRINTBASE+12    ;$790C indirizzo di GetDimensions
PrintASCII      =   PRINTBASE+15    ;$790F indirizzo di PrintASCII
StartASCII      =   PRINTBASE+18    ;$7912 indirizzo di StartASCII
SetNLQ          =   PRINTBASE+21    ;$7915 indirizzo di SetNLQ

;Indirizzi degli entry point nel driver di input

InitMouse       =   MOUSE_JMP       ;$FE80 indirizzo di InitMouse
SlowMouse       =   MOUSE_JMP+3     ;$FE83 indirizzo di SlowMouse
UpdateMouse     =   MOUSE_JMP+6     ;$FE86 indirizzo di UpdateMouse
SetMouse        =   MOUSE_JMP+9     ;$FE89 indirizzo di SetMouse (vale solo per i
                                    ;driver del C-128, DJD 2-9-87)

;Indirizzi di particolari dati per gli sprite

spr0pic         =   SPRITE_PICS     ;$8A00 indirizzo dei dati grafici dello sprite
spr1pic         =   spr0pic+64      ;$8A40 indirizzo dei dati grafici dello sprite
spr2pic         =   spr1pic+64      ;$8A80 indirizzo dei dati grafici dello sprite
spr3pic         =   spr2pic+64      ;$8AC0 indirizzo dei dati grafici dello sprite
spr4pic         =   spr3pic+64      ;$8B00 indirizzo dei dati grafici dello sprite
spr5pic         =   spr4pic+64      ;$8B40 indirizzo dei dati grafici dello sprite
spr6pic         =   spr5pic+64      ;$8B80 indirizzo dei dati grafici dello sprite
spr7pic         =   spr6pic+64      ;$8BC0 indirizzo dei dati grafici dello sprite
NMI_VECTOR      =   $FFFA           ;Locazione del vettore NMI
RESET_VECTOR    =   $FFFC           ;Locazione del vettore RESET
IRQ_VECTOR      =   $FFFE           ;Locazione del vettore IRQ
```

```
; ****************************************************************************************
;                      Definizioni e variabili del chip grafico VIC II
; ****************************************************************************************

mob0xpos        =   vicbase           ;$D000 coordinate x e y di ogni sprite
mob0ypos        =   $D001
mob1xpos        =   $D002
mob1ypos        =   $D003
mob2xpos        =   $D004
mob2ypos        =   $D005
mob3xpos        =   $D006
mob3ypos        =   $D007
mob4xpos        =   $D008
mob4ypos        =   $D009
mob5xpos        =   $D00A
mob5ypos        =   $D00B
mob6xpos        =   $D00C
mob6ypos        =   $D00D
mob7xpos        =   $D00E
mob7ypos        =   $D00F

msbxpos         =   $D010             ;Bit 9 della coordinata x di ogni sprite
grcntrl1        =   $D011             ;Registro di controllo grafico

st_ecm          =   $40               ;Bit definiti per essere utilizzati con il
st_bmm          =   $20               ;registro grcntrl1
st_den          =   $10
st_25row        =   $08

rasreg          =   $D012             ;Registro di controllo del raster
lpxpos          =   $D013             ;Coordinata x della penna ottica
lpypos          =   $D014             ;Coordinata y della penna ottica
mobenble        =   $D015             ;Controllo abilitazione degli sprite
grcntrl2        =   $D016             ;Secondo registro di controllo grafico

st_mcm          =   $10               ;Bit definiti per essere impiegati con il
st_40col        =   $08               ;registro grcntrl1
moby2           =   $D017             ;Ingrandimento verticale dello sprite
```

```
grmemptr        =  $D018              ;Puntatori grafici in memoria, per esempio
                                      ;l'inizio della matrice di schermo e della ROM
                                      ;caratteri
grirq           =  $D019              ;Registro di controllo degli interrupt grafici
grirqen         =  $D01A              ;Registro di abilitazione degli interrupt grafici
st_rasen        =  $01                ;Bit per abilitare l'interrupt di raster
                                      ;in grirqen
mobprior        =  $D01B              ;Priorita' degli sprite con lo sfondo
mobmcm          =  $D01C              ;Opzione multicolor per gli sprite
mobx2           =  $D01D              ;Ingrandimento orizzontale dello sprite
mobmobcol       =  $D01E              ;Registro di collisione fra sprite
mobbakcol       =  $D01F              ;Registro di collisione fra sprite e sfondo
extclr          =  $D020              ;Colore del bordo
backlr0         =  $D021              ;Colore 0 di sfondo
backlr1         =  $D022              ;Colore 1 di sfondo
backlr2         =  $D023              ;Colore 2 di sfondo
backlr3         =  $D024              ;Colore 3 di sfondo
mcmclr0         =  $D025              ;Colore 0 degli sprite in multicolor
mcmclr1         =  $D026              ;Colore 1 degli sprite in multicolor
mob0clr         =  $D027              ;Colori dello sprite
mob1clr         =  $D028              ;Colori dello sprite
mob2clr         =  $D029              ;Colori dello sprite
mob3clr         =  $D02A              ;Colori dello sprite
mob4clr         =  $D02B              ;Colori dello sprite
mob5clr         =  $D02C              ;Colori dello sprite
mob6clr         =  $D02D              ;Colori dello sprite
mob7clr         =  $D02E              ;Colori dello sprite
keyreg          =  $D02F              ;Registro del C-128 di controllo della tastiera
clkreg          =  $D030              ;Registro del C-128 per la frequenza di clock
                                      ;(1 o 2 MHz)

obj0Pointer     =  COLOR_MATRIX+$03F8 ;Puntatori ai disegni degli sprite in RAM
obj1Pointer     =  $8FF9
obj2Pointer     =  $8FFA
obj3Pointer     =  $8FFB
obj4Pointer     =  $8FFC
obj5Pointer     =  $8FFD
obj6Pointer     =  $8FFE
obj7Pointer     =  $8FFF
```

```
; ****************************************************************************************
;            Definizioni e variabili per il chip SID (6581) generatore di suoni
; ****************************************************************************************

v1freqlo        =   sidbase          ;$D400, registri solo in scrittura
v1freqhi        =   $D401
v1pwlo          =   $D402
v1pwhi          =   $D403
v1cntrl         =   $D404
v1attdec        =   $D405
v1susrel        =   $D406
v2freqlo        =   $D407
v2freqhi        =   $D408
v2pwlo          =   $D409
v2pwhi          =   $D40A
v2cntrl         =   $D40B
v2attdec        =   $D40C
v2susrel        =   $D40D
v3freqlo        =   $D40E
v3freqhi        =   $D40F
v3pwlo          =   $D410
v3pwhi          =   $D411
v3cntrl         =   $D412
v3attdec        =   $D413
v3susrel        =   $D414
fclo            =   $D415
fchi            =   $D416
resfilt         =   $D417
modevol         =   $D418
potx            =   $D419            ;Registri solo in lettura
poty            =   $D41A
osc3rand        =   $D41B
env3            =   $D41C
```

```
; *******************************************************************************************
;         Definizioni e variabili per il chip CIA (6526) interfaccia di comunicazione
; *******************************************************************************************


cia1pra         =    cia1base          ;$DC00 Registro dati 'a' del cia1
cia1prb         =    $DC01             ;Registro dati 'b' del cia1
cia1ddra        =    $DC02             ;Registro direzione dati 'a' del cia1
cia1ddrb        =    $DC03             ;Registro direzione dati 'b' del cia1
cia1talo        =    $DC04             ;Registro basso del timer 'a' del cia1
cia1tahi        =    $DC05             ;Registro alto del timer 'a' del cia1
cia1 lo         =    $DC06             ;Registro basso del timer 'b' del cia1
cia1 hi         =    $DC07             ;Registro alto del timer 'b' del cia1
cia1tod10ths    =    $DC08             ;Registro dei decimi di secondo del cia1
cia1todsec      =    $DC09             ;Registro dei secondi del cia1
cia1todmin      =    $DC0A             ;Registro dei minuti del cia1
cia1todhr       =    $DC0B             ;Registro delle ore AM/PM del cia1
cia1sdr         =    $DC0C             ;Registro dati seriali del cia1
cia1icr         =    $DC0D             ;Registro di controllo degli interrupt del cia1
cia1cra         =    $DC0E             ;Registro 'a' di controllo del cia1
cia1crb         =    $DC0F             ;Registro 'b' di controllo del cia1

cia2pra         =    cia2base          ;$DD00 Registro dati 'a' del cia2
cia2prb         =    $DD01             ;Registro dati 'b' del cia2
cia2ddra        =    $DD02             ;Registro direzione dati 'a' del cia2
cia2ddrb        =    $DD03             ;Registro direzione dati 'b' del cia2
cia2talo        =    $DD04             ;Registro basso del timer 'a' del cia2
cia2tahi        =    $DD05             ;Registro alto del timer 'a' del cia2
cia2 lo         =    $DD06             ;Registro basso del timer 'b' del cia2
cia2 hi         =    $DD07             ;Registro alto del timer 'b' del cia2
cia2tod10ths    =    $DD08             ;Registro dei decimi di secondo del cia2
cia2todsec      =    $DD09             ;Registro dei secondi del cia2
cia2todmin      =    $DD0A             ;Registro dei minuti del cia2
cia2todhr       =    $DD0B             ;Registro delle ore AM/PM del cia2
cia2sdr         =    $DD0C             ;Registro dati seriali del cia2
cia2icr         =    $DD0D             ;Registro di controllo degli interrupt del cia2
cia2cra         =    $DD0E             ;Registro 'a' di controllo del cia2
cia2crb         =    $DD0F             ;Registro 'b' di controllo del cia2
```

```
;Pagina 0

;00-01           Registri di I/O del 6510
;02-33           Pseudoregistri da due byte chiamati r0-r15. Vengono utilizzati per
                 ;passare parametri fra le applicazioni e il Kernel di GEOS, e memorizzare
                 ;temporaneamente i dati. Questi registri vengono salvati all'inizio
                 ;di una procedura di interrupt (InterruptMain) in modo che sia MainLoop
                 ;(o l'applicazione) sia InterruptMain possano utilizzare le stesse
                 ;routine "contemporaneamente"
;34-6F           Variabili globali del sistema operativo
;70-7F           Registri riservati per le applicazioni a2-a9
;80-FA           Spazio correntemente riservato al Kernel del C-64
;FB-FE           Registri riservati per le applicazioni a0-a1. Questo spazio di memoria e'
                 ;completamente disponibile e non viene alterato da operazioni di sistema
;FF              Utilizzato dal Kernel del C-64 e dal Basic


;Pseudoregistri

                 .zsec           $00
zpage:           .block 2        ;Registri del 6510

r0:              .block 2
r0L              =               $02
r0H              =               $03
r1:              .block 2
r1L              =               $04
r1H              =               $05
r2:              .block 2
r2L              =               $06
r2H              =               $07
r3:              .block 2
r3L              =               $08
r3H              =               $09
r4:              .block 2
r4L              =               $0A
r4H              =               $0B
r5:              .block 2
r5L              =               $0C
r5H              =               $0D
```

```
r6:             .block 2
r6L             =               $0E
r6H             =               $0F
r7:             .block 2
r7L             =               $10
r7H             =               $11
r8:             .block 2
r8L             =               $12
r8H             =               $13
r9:             .block 2
r9L             =               $14
r9H             =               $15
r10:            .block 2
r10L            =               $16
r10H            =               $17
r11:            .block 2
r11L            =               $18
r11H            =               $19
r12:            .block 2
r12L            =               $1A
r12H            =               $1B
r13:            .block 2
r13L            =               $1C
r13H            =               $1D
r14:            .block 2
r14L            =               $1E
r14H            =               $1F
r15:            .block 2
r15L            =               $20
r15H            =               $21

;Tutte le variabili che iniziano a questa locazione vengono salvate durante
;l'esecuzione di un accessorio da scrivania (Desk Accessory) e l'apertura di un BD

s_zp_global:

;Questa variabile locale e' per le routine grafiche

curPattern:     .block 2        ;$0022 puntatore alla matrice grafica corrente

;Questa variabile viene utilizzata da GetString per memorizzare l'indirizzo
;del buffer che l'applicazione ha allocato per ricevere la stringa di input
```

```
string:          .block 2       ;$0024

;Variabili globali utilizzate dai codici di gestione delle fonti caratteri per definire
;il set di caratteri

cardData:
baselineOffset:  .block 1       ;$0026 offset dal lato superiore alla linea di base
                                ;nel set di caratteri
curSetWidth:     .block 2       ;$0027 larghezza in byte di una linea del set (somma
                                ;orizzontale delle larghezze di tutti i
                                ;caratteri nel set)
curHeight:       .block 1       ;$0029 corpo carattere del set (altezza in pixel,
                                ;numero di linee o point size del set)
curIndexTable:   .block 2       ;$002A offset dal primo byte della fonte su disco
                                ;all'inizio della tavola indice contenente gli
                                ;offset in pixel per ogni carattere del set (questi
                                ;offset sono relativi all'inizio di ogni linea del
                                ;set)
cardDataPntr:    .block 2       ;$002C puntatore all'inizio della prima linea del
                                ;set in memoria; l'indirizzo delle altre linee
                                ;viene calcolato da GEOS aggiungendo l'offset
                                ;memorizzato in curSetWidth

endCardData:

CARD_DATA_ITEMS  = endCardData - cardData

;Modo corrente (stile) di visualizzazione dei caratteri

;                %1xxxxxxx      underline (sottolineato)
;                %0xxxxxxx      no underline
;                %x1xxxxxx      italic (corsivo)
;                %x0xxxxxx      no italic
;                %xx1xxxxx      reverse (negativo)
;                %xx0xxxxx      no reverse

currentMode:     .block 1       ;$002E modo corrente di scrittura

;Questo flag controlla lo stato di attivita' dello schermo principale e di quello nascosto

dispBufferOn:    .block 1       ;$002F bit 7 - controlla l'accesso allo schermo principale
                                ;      bit 6 - controlla l'accesso allo schermo nascosto
```

```
;Questa variabile globale indica lo stato del mouse, dei menu e delle icone

mouseOn:        .block 1        ;$0030          bit 7 - mouse on(1)/off(0)
                                ;               bit 6 - menu on(1)/off(0)
                                ;               bit 5 - icone on(1)/off(0)

msePicPtr:      .block 2        ;$0031 puntatore ai dati grafici del mouse

;Queste variabili globali vengono utilizzate dalle routine grafiche e di gestione
;dei testi per mascherare. Nella versione 1.1 solo le routine di gestione dei testi riconoscono
;queste variabili

windowTop:      .block 1        ;$0033 linea superiore della finestra oltre la quale viene
                                ;operato il mascheramento dei testi
windowBottom:   .block 1        ;$0034 linea inferiore della finestra oltre la quale viene
                                ;operato il mascheramento dei testi
leftMargin:     .block 2        ;$0035 punto piu' a sinistra per visualizzare i caratteri
                                ;CR posiziona il cursore a questa distanza dal lato
                                ;sinistro dello schermo
rightMargin:    .block 2        ;$0037 punto piu' a destra per visualizzare i caratteri
                                ;se viene oltrepassato, GEOS esegue la routine puntata dal
                                ;vettore stringFaultVec

;Qui finisce la zona di pagina 0 che GEOS salva durante l'esecuzione di un accessorio
;da scrivania (Desk Accessory) o durante l'apertura di un BD (Box di Dialogo)

e_zp_global:

;Il flag che segue indica lo stato del mouse e se e' stato premuto un tasto

pressFlag:      .block 1        ;$0039
                                ;bit 7 - impostato a 1 da InterruptMain se l'utente ha
                                ;        premuto un tasto sulla tastiera
                                ;bit 6 - il mouse ha cambiato il suo stato
                                ;bit 5 - il pulsante del mouse ha cambiato il suo stato

;Posizioni del Mouse

mouseXPos:      .block 2        ;$003A coordinata x del mouse
mouseYPos:      .block 1        ;$003C coordinata y del mouse
```

```
;Questa variabile viene utilizzata da GEOS per le routine che prevedono la chiamata inline
returnAddress:    .block 2      ;$003D indirizzo al quale ritornare dopo la chiamata inline
                                ;a una routine. Viene impiegata da DoInlineReturn

graphicsMode:     .block 1      ;$003F $00 per VIC (C-64)
                                ;$80 per VDC 640 x 200 (C-128)
                                ;$C0 per VDC 640 x 400 (C-128, ma non attivabile)


; ****************************************************************************************
;         Spazio in pagina 0 per le applicazioni, che NON dev'essere usato da GEOS e
;                                dagli accessori da scrivania
; ****************************************************************************************


a0                =             $FB
a0L               =             $FB
a0H               =             $FC
a1                =             $FD
a1L               =             $FD
a1H               =             $FE
a2                =             $70
a2L               =             $70
a2H               =             $71
a3                =             $72
a3L               =             $72
a3H               =             $73
a4                =             $74
a4L               =             $74
a4H               =             $75
a5                =             $76
a5L               =             $76
a5H               =             $77
a6                =             $78
a6L               =             $78
a6H               =             $79
a7                =             $7A
a7L               =             $7A
a7H               =             $7B
a8                =             $7C
a8L               =             $7C
a8H               =             $7D
```

```
a9              =           $7E
a9L             =           $7E
a9H             =           $7F


; ******************************************************************************************
;                        Inizio della RAM di sistema impiegata da GEOS
; ******************************************************************************************

;Nota: quest'area della memoria inizia con i buffer da 256 byte usati durante gli accessi
;al disco. Se vengono riallocati, devono sempre iniziare con il primo byte di una pagina
;della memoria

                .ramsect        OS_VARS         ;$8000

diskBlkBuf:     .block 256      ;$8000 buffer per contenere un qualunque blocco
                                ;del disco
fileHeader:     .block 256      ;$8100 buffer per contenere il File Header Block
                                ;di un file GEOS
curDirHead:     .block 256      ;$8200 buffer per contenere il Directory Header Block
                                ;del disco
fileTrScTab:    .block 256      ;$8300 buffer utilizzato per contenere gli indirizzi T/S dei
                                ;settori che realizzano il concatenamento del file
                                ;Dal momento che normalmente il primo indirizzo T/S
                                ;individua il blocco File Header del file, il numero
                                ;di indirizzi non puo' essere superiore a 127 e quindi
                                ;il file non puo' essere piu' lungo di 32.258 byte. Se si
                                ;deve salvare sul disco un file di dimensioni maggiori,
                                ;possono essere utilizzate le routine di
                                ;lettura/scrittura parziale, e una speciale tavola
                                ;di indirizzi T/S di dimensioni sufficienti

startRamZero:                   ;Inizio della RAM che viene automaticamente azzerata durante
                                ;l'installazione del sistema (booting). Da questo indirizzo
                                ;fino a $88FF le locazioni di memoria vengono azzerate

dirEntryBuf:    .block 30       ;$8400 buffer utilizzato per creare il File Entry di un file
```

```
; ****************************************************************************************
;                              Variabili per gli accessi al disco
; ****************************************************************************************

;I due buffer che seguono contengono il nome del disco corrente contenuto in uno dei due
;possibili disk drive

DrACurDkNm:      .block 18               ;$841E nome del disco nel drive A
                                         ;16 caratteri per il nome + 2 caratteri per l'ID
DrBCurDkNm:      .block 18               ;$8430 nome del disco nel drive B
                                         ;16 caratteri per il nome + 2 caratteri per l'ID
dataFileName:    .block 17               ;$8442 nome del file dati che dev'essere passato
                                         ;all'applicazione
dataDiskName:    .block 18               ;$8453 nome del disco sul quale e' memorizzato il
                                         ;file dati
prntFileName:    .block 17               ;$8465 nome del driver di stampa selezionato
                                         ;16 byte per il nome + 1 terminatore
prntDiskName:    .block DK_NM_ID_LEN + 1 ;$8476 nome del disco sul quale e' residente il
                                         ;driver di stampa selezionato
                                         ;18 byte per il nome del disco + 1 terminatore
curDrive:        .block 1                ;$8489 drive correntemente attivo (8, 9, 10 o 11)
diskOpenFlg:     .block 1                ;$848A indica se un disco e' aperto
isGEOS:          .block 1                ;$848B questo flag indica se il disco
                                         ;e' in formato GEOS
interleave:      .block 1                ;$848C BlkAlloc utilizza questo valore quando
                                         ;seleziona i settori del disco per creare il
                                         ;concatenamento richiesto. GEOS inizializza
                                         ;questa variabile a 8
numDrives:       .block 1                ;$848D numero dei drive attivi collegati al
                                         ;sistema
driveType:       .block 4                ;$848E tipo del disk drive (questa variabile non
                                         ;viene attualmente utilizzata)
                                         ;1 byte per ogni drive possibile: 8, 9, 10 ,11
                                         ;Le costanti per queste variabili sono riportate
                                         ;nell'appendice A come "DRV_1541"
turboFlags:      .block 4                ;$8492 flag di stato del turbo (attualmente sono
                                         ;utilizzati solo i due primi byte, dal momento che
                                         ;il sistema si puo' configurare solo per due drive)
                                         ;Flag per il turbo dei drive 8, 9, 10, 11
```

```
                                        ;Bit 6:  0 turbo residente nel drive, ma
                                        ;          correntemente disattivato
                                        ;        1 turbo in esecuzione
                                        ;Bit 7:  0 turbo non residente
                                        ;          nel drive
                                        ;        1 turbo residente nel drive

;Variabili mantenute da GEOS per contenere le informazioni sul file SEQUENTIAL o VLIR
;correntemente aperto

curRecord:      .block 1                ;$8496 record corrente
usedRecords:    .block 1                ;$8497 numero dei record nel file aperto
fileWritten:    .block 1                ;$8498 flag per indicare se il file e' stato
                                        ;aggiornato dopo l'ultima modifica effettuata
                                        ;nella tavola indice e nella BAM
fileSize:       .block 2                ;$8499 dimensione corrente (in blocchi) del file
                                        ;Questo parametro viene letto e scritto nel File
                                        ;Entry del file

;Le variabili che seguono vengono temporaneamente salvate durante l'apertura di un BD o
;l'esecuzione di un accessorio da scrivania

s_nonzp_global:


; ***************************************************************************************
;                                         Vettori
; ***************************************************************************************

appMain:        .block 2                ;$849B codice MainLoop dell'applicazione
                                        ;Permette all'applicazione di includere codici
                                        ;propri alla fine del MainLoop di sistema
intTopVector:   .block 2                ;$849D chiamata all'inizio della routine di
                                        ;interrupt di sistema
                                        ;permette alle applicazioni di includere i loro
                                        ;codici di interrupt all'interno della routine
                                        ;InterruptMain di sistema
intBotVector:   .block 2                ;$849F questo vettore viene eseguito quando
                                        ;termina la routine di interrupt di sistema. In
                                        ;questo modo le applicazioni possono aggiungere
                                        ;codici propri alla fine della routine di
                                        ;interrupt di sistema
```

```
mouseVector:      .block 2          ;$84A1 routine eseguita quando viene premuto il
                                    ;pulsante del mouse
keyVector:        .block 2          ;$84A3 routine eseguita quando viene premuto un
                                    ;tasto della tastiera
inputVector:      .block 2          ;$84A5 routine eseguita qualora avvengano dei
                                    ;cambiamenti nel dispositivo di input
mouseFaultVec:    .block 2          ;$84A7 routine eseguita quando il mouse esce dai
                                    ;limiti della regione delimitata o da un menu
otherPressVec:    .block 2          ;$84A9 routine eseguita quando il pulsante del
                                    ;mouse viene premuto su una regione dello schermo
                                    ;non convenzionale (convenzionale: menu e icone)
stringFaultVec:   .block 2          ;$84AB routine da chiamare nel caso che i caratteri
                                    ;visualizzati sullo schermo oltrepassino
                                    ;il margine destro individuato dalla variabile
                                    ;rightMargin
alarmTmtVector:   .block 2          ;$84AD routine da eseguire quando entra in
                                    ;funzione l'allarme interno del C-64. L'allarme
                                    ;viene impostato dall'applicazione. Grazie a
                                    ;questa routine puo' essere generato un suono
                                    ;particolare, oppure un segnale luminoso sullo
                                    ;schermo
BRKVector:        .block 2          ;$84AF routine da eseguire quando il processore
                                    ;si trova a eseguire l'istruzione BRK
recoverVector:    .block 2          ;$84B1 routine che viene eseguita per ripristina-
                                    ;re lo schermo alla chiusura di un menu, di un
                                    ;box di dialogo o di un desk accessory

;La variabile che segue determina la velocita' alla quale si invertono le icone e
;le opzioni dei menu quando vengono selezionate

selectionFlash:   .block 1          ;$84B3

;La variabile che segue viene utilizzata durante l'input di stringhe

alphaFlag:        .block 1          ;$84B4 alphaFlag contiene alcuni bit che governa-
                                    ;no il cursore visibile. Se il bit 7 e' impostato a
                                    ;1, il cursore visibile e' attivato. In questo caso
                                    ;il bit 6 mantiene lo stato del cursore visibile.
                                    ;Mentre il cursore lampeggia, il bit 6 indica se e'
                                    ;acceso o spento. Quando e' impostato a 0 il cursore
                                    ;visibile e' spento. I 6 bit rimanenti
                                    ;costituiscono un contatore per determinare il
                                    ;ritmo del lampeggio
```

```
;La variabile che segue contiene due bit, bit 7 e bit 6, che specificano in che modo GEOS
;segnala all'utente la selezione di un'icona. Se nessuno dei due e' impostato a 1,
;il sistema non compie alcuna azione particolare per segnalare all'utente la selezione di
;un'icona, ed esegue semplicemente la routine di servizio associata. I due flag possibili per
;questa variabile sono: ST_FLASH=$80, inverte momentaneamente l'icona selezionata
;ST_INVERT=$40, inverte in modo permanente l'icona selezionata Se il bit individuato da
;ST_FLASH e' impostato a 1, il bit 6 viene ignorato e l'icona si inverte per una frazione di
;secondo quando viene selezionata. Se il bit individuato da ST_INVERT e' impostato a 1 e
;il bit 7 e' impostato a 0, l'icona viene invertita permanentemente

iconSetFlag:      .block 1               ;$84B5

;Questa variabile riporta gli errori di spostamento del mouse

faultData:        .block 1               ;$84B6 bit di flag per gli errori del mouse,
                                         ;questa variabile viene normalmente controllata
                                         ;dalla routine puntata dal vettore mouseFaultVec
                                         ;I possibili valori che puo' assumere sono
                                         ;documentati nell'appendice A


; ***************************************************************************************
;                    Variabili globali relative alla gestione del mouse
; ***************************************************************************************

menuNumber:       .block 1               ;$84B7 numero di menu attivi

;Quando il mouse oltrepassa uno dei limiti imposti dalle seguenti variabili, GEOS
;manda in esecuzione il vettore mouseFaultVec

mouseTop:         .block 1               ;$84B8 massima altezza per il mouse
mouseBottom:      .block 1               ;$84B9 minima altezza per il mouse
mouseLeft:        .block 2               ;$84BA confine sinistro per il mouse
mouseRight:       .block 2               ;$84Bc confine destro per il mouse

;Variabili globali per la gestione delle stringhe in input e del cursore


stringX:          .block 2               ;$84BE coordinata x del punto in cui generare
                                         ;l'eco sullo schermo durante l'input di stringhe
stringY:          .block 1               ;$84C0 coordinata y del punto in cui generare
                                         ;l'eco sullo schermo durante l'input di stringhe
```

```
e_nonzp_global:                         ;Questa e' la fine delle variabili globali non
                                        ;residenti in pagina 0 che vengono salvate durante
                                        ;l'apertura dei box di dialogo e l'esecuzione
                                        ;degli accessori da scrivania


mousePicData:     .block 64             ;$84C1 matrice in RAM che contiene i dati grafici
                                        ;del disegno del mouse
maxMouseSpeed:    .block 1              ;$8501 velocita' massima di spostamento del mouse
minMouseSpeed:    .block 1              ;$8502 velocita' minima di spostamento del mouse
mouseaccel:       .block 1              ;$8503 accelerazione del mouse

;Le variabili globali che seguono mantengono lo stato corrente del mouse e della tastiera


keyData:          .block 1              ;$8504 questa e' la variabile che contiene il
                                        ;codice del tasto premuto. La routine di servizio
                                        ;della tastiera deve accedere a questa variabile
mouseData:        .block 1              ;$8505 questa e' la variabile alla quale la routine
                                        ;di servizio del mouse deve accedere per leggere lo
                                        ;stato del dispositivo di input
inputData:        .block 4              ;$8506 queste 4 variabili sono a disposizione del
                                        ;driver di input per passare informazioni speci-
                                        ;fiche per il particolare dispositivo di input

;Numero random. Viene incrementato a ogni chiamata di interrupt


random:           .block 2              ;$850A
saveFontTab:      .block 9              ;$850C quando si accede ai menu, si deve salvare la
                                        ;tavola di definizione della fonte corrente in
                                        ;questo buffer

;La variabile che segue viene impiegata per registrare la doppia pressione del pulsante
;del mouse sopra un'icona
;dblClickCount viene aggiornata con un particolare valore quando l'icona viene selezionata
;con la prima pressione del pulsante del mouse
;Dal momento che in questo periodo di tempo la variabile contiene un valore diverso
;da zero, la routine di interrupt di sistema provvede a decrementarla a ogni chiamata
;Se quando l'icona viene nuovamente selezionata il valore in dblClickCount non e' 0,
;GEOS stabilisce che l'utente ha selezionato l'icona due volte in rapida successione, e
;restituisce alla routine di servizio il valore TRUE nel registro r0H. Se invece
```

```
;la variabile dblClickCount e' 0 quando l'utente la seleziona di nuovo, GEOS passa il
;valore FALSE in r0H

dblClickCount:    .block 1          ;$8515 viene utilizzata per determinare le doppie
                                    ;pressioni del pulsante del mouse sopra un'icona

;Variabili associate alla data e all'ora di sistema

year:             .block 1          ;$8516
month:            .block 1          ;$8517
day:              .block 1          ;$8518
hour:             .block 1          ;$8519
minutes:          .block 1          ;$851A


seconds:          .block 1          ;$851B

;La variabile globale che segue comanda l'abilitazione dell'allarme

alarmSetFlag:     .block 1          ;$851C se vale TRUE l'allarme non e' impostato
                                    ;Se vale FALSE l'allarme e' inserito

;Le variabili che seguono sono associate alla gestione dei box di dialogo


sysDBData         .block 1          ;$851D viene utilizzata internamente per
                                    ;determinare quale comando ha causato la chiusura
                                    ;del BD e il ritorno all'applicazione. Il codice
                                    ;del comando viene restituito in r0L
screenColors:     .block 1          ;$851E colore di default dello schermo


;Si calcola il valore delle costanti che seguono per decidere la dimensione del buffer
;impiegato per salvare tutte le variabili RAM che caratterizzano lo stato
;dell'applicazione. Queste costanti individuano tutta le variabili RAM specificate
;nella tavola saveSysRamTab utilizzata dal modulo SaveSysRam


MENU_SPACE        = (3 * MAXIMUM_MENU_NESTING) + (2 * MAXIMUM_MENU_ITEMS)
                                    ;$002A byte interessati
SRAM_ZP_GLOBAL    = e_zp_global - s_zp_global
                                    ;$0017 byte interessati
```

```
SRAM_NONZP_GLOBAL = e_nonzp_global - s_nonzp_global
                                     ;$0026 byte interessati
SRAM_LOCAL        = 2  +  14+MENU_SPACE  +  (MAXIMUM_PROCESSES * 8)  +  (SLEEP_MAXIMUM * 4)
                                     ;$012A byte interessati
SRAM_SPRITES      = $0026            ;byte interessati
TOT_SRAM_SAVED
                  = SRAM_ZP_GLOBAL+SRAM_NONZP_GLOBAL+SRAM_LOCAL+SRAM_SPRITES+20
                                     ;$01A1 byte interessati
dlgBoxRamBuf:  .block TOT_SRAM_SAVED ;$851F buffer per salvare tutte le variabili di sistema
                                     ;che caratterizzano l'applicazione, specificate
                                     ;nella tavola saveSysRamTab



; ********************************************************************************************
;                        Seconda area di memoria per le variabili globali
; ********************************************************************************************


;Le variabili che seguono sono raccolte in una seconda area globale di memoria. Queste
;variabili vengono utilizzate dalle applicazioni create per GEOS V1.3 e dallo stesso Kernel
;V1.3. Dal momento che l'utente puo' anche mandare in esecuzione deskTop V1.3 sotto GEOS
;V1.2, e' molto importante che queste variabili siano allocate sempre nella stessa area
;di memoria. In questo modo, per esempio, l'utente che possiede GEOS V1.2 e ha ricevuto
;deskTop V1.3 via Q-LINK, puo' ugualmente giovarsi dell'auto-booting dei driver di input
;Per ragioni di compatibilita', queste variabili devono quindi essere allocate sempre
;nella stessa area

                  .ramsect      $88BB

savedmoby2:       .block 1           ;$88BB questa variabile dev'essere aggiornata con
                                     ;il valore contenuto in moby2 durante l'apertura
                                     ;del box di dialogo e l'esecuzione dei desk
                                     ;accessory, per evitare che possa essere alterata
                                     ;Dal momento che nei codici originali di GEOS la
                                     ;variabile moby2 non viene temporaneamente
                                     ;salvata durante i BD e i DA, deve provvedere la
                                     ;stessa applicazione
scr80polar:       .block 1           ;$88BC copia del registro 24 contenuto nel VDC del
                                     ;C-128
scr80colors:      .block 1           ;$88BD colori di schermo per il modo a 80 colonne
                                     ;del C-128, copia del registro 26 del VDC
```

```
vdcClrMode:      .block 1        ;$88BE mantiene il modo colore corrente impiegato
                                 ;dalle routine del colore del C-128
driveData:       .block 2        ;$88BF 1 byte per ogni dispositivo che il disk
                                 ;driver corrente e' in grado di gestire (ogni
                                 ;driver puo' comandarne diversi)
ramExpSize:      .block 1        ;$88C3 numero di banchi di memoria
                                 ;dell'espansione installata
sysRAMFlg:       .block 1        ;$88C4 se e' presente un'espansione RAM,
                                 ;il banco 0
                                 ;viene riservato all'uso interno del Kernel
                                 ;Questa variabile indica a quale uso viene
                                 ;dedicato:
                                 ;Bit 7: se impostato a 1, $0000-$78FF impiegata
                                 ;dalla routine MoveData
                                 ;Bit 6: se impostato a 1, $8300-$B8FF contiene i
                                 ;disk driver per i drive dall'A al C
                                 ;Bit 5: se impostato a 1, $7900-$7DFF ToBasic vi
                                 ;memorizza l'area RAM di GEOS $8400-$88FF quando
                                 ;il controllo viene restituito al Basic
                                 ;Bit 4: se impostato a 1, $7E00-$82FF un file
                                 ;AUTO_EXEC di setup vi memorizza i codici di
                                 ;ricaricamento (reboot). Questi codici vengono
                                 ;ricaricati in memoria a $6000, al posto del file
                                 ;GEOS_BOOT, dal codice di restart, locato a $C000,
                                 ;se questo flag e' impostato a 1. Inoltre,
                                 ;nell'area $B900-$FC3F viene salvato il Kernel in
                                 ;maniera che possa essere installato nuovamente
                                 ;l'intero sistema senza che siano compiuti accessi
                                 ;al disco (secondo le variabili impostate dal file
                                 ;setup). Quest'area dovrebbe essere aggiornata
                                 ;quando vengono cambiati i dispositivi di input
                                 ;(operazione che viene compiuta da deskTop V1.3)
firstBoot:       .block 1        ;$88C5 questo flag viene cambiato da $00 a $FF
                                 ;quando deskTop viene caricata per la prima volta
                                 ;durante l'installazione del sistema
curType:         .block 1        ;$88C6 tipo di disco corrente (copiata da
                                 ;diskType)
ramBase:         .block 4        ;$88C7 banco di RAM dell'espansione per ogni drive
                                 ;da impiegare nel caso che il drive sia di tipo RAM
                                 ;disk o Shadowed
inputDevName:    .block 17       ;$88CB mantiene il nome del dispositivo di input
                                 ;corrente
```

```
DrCCurDkNm:      .block 18       ;$88DC nome del disco contenuto nel drive C
                                 ;18 caratteri conclusi con $A0
DrDCurDkNm:      .block 18       ;$88EE nome del disco contenuto nel drive D
                                 ;18 caratteri conclusi con $A0
EndRamZero:                      ;Fine dell'area di memoria che GEOS azzera durante
                                 ;l'installazione iniziale
dir2Head:        .block 256      ;$8900 secondo Directory Header Block, per
                                 ;dischi formattati da drive con capacita'
                                 ;maggiori (come il 1571)


                 .ramsect     EndGlobal

.end
```

APPENDICE C: ROUTINE

```
;*******************************************************************************************
;
;                                        geosRoutines
;
;Questo file contiene la tavola degli entry-point (o jump-call) a disposizione
;delle applicazioni per chiamare le routine di GEOS
;
;
;
;*******************************************************************************************



;Routine generali

;I due entry point che seguono sono gli unici memorizzati a $C000. Tutti gli altri iniziano
;da OS_JUMPTAB

BootGEOS             =     $C000      ;Ricarica GEOS integralmente, a condizione pero'
                                      ;che i codici da $C000 a $C02F non siano stati
                                      ;alterati
ResetHandle          =     $C003      ;Riattiva GEOS attraverso la procedura di Cold
                                      ;Start (partenza a freddo). Presuppone che GEOS
                                      ;sia gia' stato interamente caricato in memoria
```

```
;Gli entry point che seguono sono locati da OS_JUMPTAB in poi

InterruptMain       =     $C100
                                   ;*********************************************
                                   ;PROCESSI
                                   ;*********************************************
InitProcesses       =     $C103
RestartProcess      =     $C106
EnableProcess       =     $C109
BlockProcess        =     $C10C
UnblockProcess      =     $C10F
FreezeProcess       =     $C112
UnfreezeProcess     =     $C115
                                   ;*********************************************
                                   ;GRAFICA
                                   ;*********************************************
HorizontalLine      =     $C118
InvertLine          =     $C11B
RecoverLine         =     $C11E
VerticalLine        =     $C121
Rectangle           =     $C124
FrameRectangle      =     $C127
InvertRectangle     =     $C12A
RecoverRectangle    =     $C12D
DrawLine            =     $C130
DrawPoint           =     $C133
GraphicsString      =     $C136
SetPattern          =     $C139
GetScanLine         =     $C13C
TestPoint           =     $C13F
                                   ;*********************************************
                                   ;GENERAZIONE DI UNA SCHERMATA
                                   ;*********************************************
BitmapUp            =     $C142

                                   ;*********************************************
                                   ;MANIPOLAZIONE DEI CARATTERI
                                   ;*********************************************
PutChar             =     $C145
PutString           =     $C148
UseSystemFont       =     $C14B
```

```
                                        ;*******************************************
                                        ;MOUSE, MENU E ICONE
                                        ;*******************************************
StartMouseMode      =     $C14E
DoMenu              =     $C151
RecoverMenu         =     $C154
RecoverAllMenus     =     $C157
DoIcons             =     $C15A
                                        ;*******************************************
                                        ;UTILITY
                                        ;*******************************************
DShiftLeft          =     $C15D
BBMult              =     $C160
BMult               =     $C163
DMult               =     $C166
Ddiv                =     $C169
DSdiv               =     $C16C
Dabs                =     $C16F
Dnegate             =     $C172
Ddec                =     $C175
ClearRam            =     $C178
FillRam             =     $C17B
MoveData            =     $C17E
InitRam             =     $C181
PutDecimal          =     $C184
GetRandom           =     $C187
                                        ;*******************************************
                                        ;MOUSE, MENU, GRAFICA, SLEEP
                                        ;*******************************************
MouseUp             =     $C18A
MouseOff            =     $C18D
DoPreviousMenu      =     $C190
ReDoMenu            =     $C193
GetSerialNumber     =     $C196
Sleep               =     $C199
ClearMouseMode      =     $C19C
i_Rectangle         =     $C19F
i_FrameRectangle    =     $C1A2
i_RecoverRectangle  =     $C1A5
i_GraphicsString    =     $C1A8
```

```
                                        ;*******************************************
                                        ;GENERAZIONE DI UNA SCHERMATA
                                        ;*******************************************
i_BitmapUp          =       $C1AB
                                        ;*******************************************
                                        ;MANIPOLAZIONE DEI CARATTERI
                                        ;*******************************************
i_PutString         =       $C1AE
GetRealSize         =       $C1B1
                                        ;*******************************************
                                        ;UTILITY
                                        ;*******************************************
i_FillRam           =       $C1B4
i_MoveData          =       $C1B7
                                        ;*******************************************
                                        ;ROUTINE AGGIUNTE
                                        ;SUCCESSIVAMENTE
                                        ;*******************************************
GetString           =       $C1BA
GotoFirstMenu       =       $C1BD
InitTextPrompt      =       $C1C0
MainLoop            =       $C1C3
DrawSprite          =       $C1C6
GetCharWidth        =       $C1C9
LoadCharSet         =       $C1CC
PosSprite           =       $C1CF
EnablSprite         =       $C1D2
DisablSprite        =       $C1D5
CallRoutine         =       $C1D8
CalcBlksFree        =       $C1DB
ChkDkGEOS           =       $C1DE
NewDisk             =       $C1E1
GetBlock            =       $C1E4
PutBlock            =       $C1E7
SetGEOSDisk         =       $C1EA
SaveFile            =       $C1ED
SetGDirEntry        =       $C1F0
BldGDirEntry        =       $C1F3
GetFreeDirBlk       =       $C1F6
WriteFile           =       $C1F9
BlkAlloc            =       $C1FC
ReadFile            =       $C1FF
```

```
SmallPutChar         =    $C202
FollowChain          =    $C205
GetFile              =    $C208
FindFile             =    $C20B
CRC                  =    $C20E
LdFile               =    $C211
EnterTurbo           =    $C214
LdDeskAcc            =    $C217
ReadBlock            =    $C21A
LdApplic             =    $C21D
WriteBlock           =    $C220
VerWriteBlock        =    $C223
FreeFile             =    $C226
GetFHdrInfo          =    $C229
EnterDeskTop         =    $C22C
StartAppl            =    $C22F
ExitTurbo            =    $C232
PurgeTurbo           =    $C235
DeleteFile           =    $C238
FindFTypes           =    $C23B
RstrAppl             =    $C23E
ToBasic              =    $C241
FastDelFile          =    $C244
GetDirHead           =    $C247
PutDirHead           =    $C24A
NxtBlkAlloc          =    $C24D
ImprintRectangle     =    $C250
i_ImprintRectangle   =    $C253
DoDlgBox             =    $C256
RenameFile           =    $C259
InitForIO            =    $C25C
DoneWithIO           =    $C25F
DShiftRight          =    $C262
CopyString           =    $C265
CopyFString          =    $C268
CmpString            =    $C26B
CmpFString           =    $C26E
FirstInit            =    $C271
OpenRecordFile       =    $C274
CloseRecordFile      =    $C277
NextRecord           =    $C27A
PreviousRecord       =    $C27D
```

```
PointRecord         =    $C280
DeleteRecord        =    $C283
InsertRecord        =    $C286
AppendRecord        =    $C289
ReadRecord          =    $C28C
WriteRecord         =    $C28F
SetNextFree         =    $C292
UpdateRecordFile    =    $C295
GetPtrCurDkNm       =    $C298
PromptOn            =    $C29B
PromptOff           =    $C29E
OpenDisk            =    $C2A1
DoInLineReturn      =    $C2A4
GetNextChar         =    $C2A7
BitmapClip          =    $C2AA
FindBAMBit          =    $C2AD
SetDevice           =    $C2B0
IsMseInRegion       =    $C2B3
ReadByte            =    $C2B6

;La routine che segue, FreeBlock, e' direttamente accessibile all'indirizzo $C2B9 solo
;dalla versione 1.3 del Kernel. Per la versione 1.2 si deve utilizzare la chiamata diretta
;all'indirizzo $9844

FreeBlock           =    $C2B9
ChangeDiskDevice    =    $C2BC
RstrFrmDialogue     =    $C2BF
Panic               =    $C2C2
BitOtherClip        =    $C2C5

;Routine per la gestione delle espansioni RAM Queste routine sono disponibili solo nelle
;versioni del Kernel successive alla 1.2

StashRAM            =    $C2C8
FetchRAM            =    $C2CB
SwapRAM             =    $C2CE
VerifyRAM           =    $C2D1
DoRAMop             =    $C2D4
```

```
                                    ;*********************************************
                                    ;DRIVER DI STAMPA
                                    ;*********************************************

InitForPrint        =       $7900   ;PRINTBASE
StartPrint          =       $7903
PrintBuffer         =       $7906
StopPrint           =       $7909
GetDimensions       =       $790C
PrintASCII          =       $790F
StartASCII          =       $7912
SetNLQ              =       $7915
```

D APPENDICE D: MACRO ISTRUZIONI

In questo file sono riportate tutte le macro istruzioni per il compilatore citate nel testo. Sono state esplicitamente create per il nostro compilatore, e quindi possono non essere adatte ad altri compilatori. In questo caso è necessario trasformarle o espanderle nei listati sorgente.

```
; ***************************************************************************************
;
;  geosMacros
;
;  Il file contiene alcune definizioni di macro che possono
;  essere impiegate nella realizzazione delle applicazioni
;
; ***************************************************************************************


; ***************************************************************************************
;
;  Load Byte:       LoadB dest, valore
;
;  Argomenti:       dest          indirizzo del byte nel quale dev'essere
;                                 memorizzato "valore"
;                   valore        valore del byte da memorizzare
;
;  Funzione:        Memorizza "valore" nel byte dest
;
; ***************************************************************************************
```

```
.macro   LoadB       dest, valore
         lda         #valore               ;carica valore
         sta         dest                  ;e lo memorizza
.endm


; ****************************************************************************************
;
;  Load Word:        LoadW dest, valore
;
;  Argomenti:        dest                    indirizzo della word nella quale
;                                            dev'essere memorizzato "valore"
;                    valore                  word da memorizzare
;
;  Funzione:         Memorizza "valore" nella word dest
;
; ****************************************************************************************

.macro   LoadW       dest, valore
         lda         #>(valore)            ;preleva il byte alto di "valore"
         sta         dest + 1              ;e lo memorizza
         lda         #<(valore)            ;preleva il byte basso di "valore"
         sta         dest + 0              ;e lo memorizza
.endm


; ****************************************************************************************
;
;  Move Byte:        MoveB sorg, dest
;
;  Argomenti:        sorg                    indirizzo sorgente
;                    dest                    indirizzo destinazione
;
;  Funzione:         Muove il contenuto del byte sorg in dest
;
; ****************************************************************************************

.macro  MoveB        sorg, dest
        lda          sorg                  ;preleva il valore dal byte sorg
        sta          dest                  ;e lo memorizza in dest
.endm
```

```
; ******************************************************************************************
;
;  Move Word:       MoveW sorg, dest
;
;  Argomenti:       sorg                     indirizzo sorgente
;                   dest                     indirizzo destinazione
;
;  Funzione:        Muove il contenuto di una word
;                   dall'indirizzo sorg all'indirizzo dest
;
; ******************************************************************************************

.macro  MoveW       sorg, dest
        lda         sorg + 1             ;preleva il byte alto
        sta         dest + 1             ;e lo memorizza
        lda         sorg + 0             ;preleva il byte basso
        sta         dest + 0             ;e lo memorizza
.endm


; ******************************************************************************************
;
;  Add Byte:        add sorg
;
;  Argomenti:       sorg                     indirizzo del byte da sommare, o
;                                            valore immediato da sommare
;
;  Funzione:        a = a + sorg
;
; ******************************************************************************************

.macro add          sorg
       clc
       adc          sorg
.endm
```

```
; ********************************************************************************************
;
;  Add Bytes:        AddB sorg, dest
;
;  Argomenti:        sorg                    indirizzo del byte da sommare
;                    dest                    indirizzo del byte da sommare che riceve
;                                            il risultato
;
;  Funzione:         dest = dest + sorg
;
; ********************************************************************************************

.macro  AddB         sorg, dest
        clc                                 ;la somma avviene senza riporto iniziale
        lda          sorg                   ;preleva il byte sorg
        adc          dest                   ;lo somma al byte dest
        sta          dest                   ;memorizza il risultato
.endm


; ********************************************************************************************
;
;  Add Words:        AddW sorg, dest
;
;  Argomenti:        sorg                    indirizzo della word da sommare
;                    dest                    indirizzo della word destinazione da sommare
;
;  Funzione:         dest = dest + sorg
;
; ********************************************************************************************

.macro  AddW         sorg, dest
        lda          sorg                   ;preleva il byte basso da sorg
        clc
        adc          dest + 0               ;lo somma al byte basso destinazione
        sta          dest + 0               ;memorizza il risultato, e riporta
                                            ;il carry
        lda          sorg + 1               ;preleva il byte alto da sorg
        adc          dest + 1               ;lo somma al byte alto destinazione
        sta          dest + 1               ;memorizza il risultato
.endm
```

```
; *****************************************************************************************
;
;  Add Value To Byte:          AddVB valore, dest
;
;  Argomenti:        valore                  costante da sommare a dest
;                    dest                    indirizzo del byte dest
;
;  Funzione:         dest = dest + valore
;
; *****************************************************************************************

.macro  AddVB        valore, dest
        lda          dest
        clc
        adc          #valore
        sta          dest
.endm

; *****************************************************************************************
;
;  Add Value To Word:          AddVW valore, dest
;
;  Argomenti:        valore                  costante da sommare a dest
;                    dest                    indirizzo della word dest
;
;  Funzione:         dest = dest + valore
;
; *****************************************************************************************

.macro  AddVW        valore, dest
        clc                             ;inizialmente nessun riporto
        lda          #<(valore)         ;preleva il byte basso di valore
        adc          dest + 0           ;e lo somma al byte basso della word
        sta          dest + 0           ;memorizza il risultato

.if                  (valore >= 0) && (valore <= 255)
        bcc          noInc              ;c'e' riporto se la somma ha prodotto
                                        ;overflow
        inc          dest + 1           ;incrementa il byte alto della word
noInc:
.else
        lda          #>(valore)         ;c'e' riporto se la somma ha prodotto
                                        ;overflow
```

```
        adc         dest + 1                ;somma il carry e il byte alto di valore
        sta         dest + 1                ;memorizza il risultato
.endif

.endm


; ************************************************************************************
;
;  Subtract Byte:   sub sorg
;
;  Argomenti:       sorg                     indirizzo del byte da sottrarre, o
;                                            valore immediato da sottrarre
;
;  Funzione:        a = a - sorg
;
; ************************************************************************************

.macro  sub         sorg
        sec
        sbc         sorg
.endm


; ************************************************************************************
;
;  Sub Bytes:       SubB sorg, dest
;
;  Argomenti:       sorg                     indirizzo del byte da sottrarre
;                   dest                     indirizzo del byte dal quale sottrarre
;
;  Funzione:        dest = dest - sorg
;
; ************************************************************************************

.macro  SubB        sorg, dest
        sec                                 ;riporto a 1 per iniziare
        lda         dest                    ;preleva il byte destinazione
        sbc         sorg                    ;sottrae il byte sorgente
        sta         dest                    ;e memorizza il risultato
.endm
```

```
; ******************************************************************************************
;
;  Sub Words:       SubW sorg, dest
;
;  Argomenti:       sorg                      indirizzo della word da sottrarre
;                   dest                      indirizzo della word dalla quale sottrarre
;
;  Funzione:        dest = dest - sorg
;
; ******************************************************************************************

.macro  SubW        sorg, dest
        lda         dest + 0             ;preleva il byte basso di dest
        sec
        sbc         sorg + 0             ;gli sottrae il byte basso di sorg
        sta         dest + 0             ;e lo memorizza
        lda         dest + 1             ;preleva il byte alto di dest
        sbc         sorg + 1             ;sottrae il byte alto di sorg
                                         ;con il riporto
        sta         dest + 1             ;memorizza il risultato
.endm


; ******************************************************************************************
;
;  Compare Bytes:   CmpB sorg, dest
;
;  Argomenti:       sorg                      indirizzo del primo byte
;                   dest                      indirizzo del secondo byte
;
;  Funzione:        Confronta il contenuto del byte
;                   con quello destinazione
;
; ******************************************************************************************

.macro  CmpB        sorg, dest
        lda         sorg                 ;preleva il byte sorgente
        cmp         dest                 ;lo confronta con il byte destinazione
.endm
```

```
; ***************************************************************************************
;
;  Compare Byte to Value:  CmpBI sorg, immed
;
;  Argomenti:       sorg                    indirizzo del primo byte
;                   immed                   valore immediato da confrontare
;
;  Funzione:        Confronta il contenuto di sorg con il
;                   valore immediato
;
; ***************************************************************************************

.macro  CmpBI       sorg, immed
        lda         sorg                  ;preleva il byte sorgente
        cmp         #immed                ;confronta con il valore immediato
.endm


; ***************************************************************************************
;
;  Compare Words:   CmpW sorg, dest
;
;  Argomenti:       sorg                    indirizzo della prima word
;                   dest                    indirizzo della seconda word
;
;  Funzione:        Confronta il contenuto della word sorg con
;                   quello della word dest
;
; ***************************************************************************************

.macro  CmpW        sorg, dest
        lda         sorg + 1              ;preleva il byte alto di sorg
        cmp         dest + 1              ;lo confronta con dest
        bne         done                  ;si deve confrontare anche il byte basso?
        lda         sorg + 0              ;esegue il confronto sul byte basso
        cmp         dest + 0              ;confronta
done:
.endm
```

```
; *********************************************************************************************
;
;  Compare Word To Value:   CmpWI sorg, immed
;
;  Argomenti:       sorg                      indirizzo della prima word
;                   immed                     valore immediato da confrontare
;
;  Funzione:        Confronta la word sorg con il valore immediato
;
; *********************************************************************************************

.macro  CmpWI       sorg, immed
        lda         sorg + 1                ;preleva il byte alto
        cmp         #>(immed)               ;confronta con il byte alto di immed
        bne         done                    ;si deve confrontare anche il byte basso?
        lda         sorg + 0                ;preleva il byte basso
        cmp         #<(immed)               ;confronta con il byte basso di immed
done:
.endm


; *********************************************************************************************
;
;  Push Byte:       PushB sorg
;
;  Argomenti:       sorg                      indirizzo del byte da inserire nello
;                                             stack
;
;  Funzione:        Inserisce nello stack il byte all'indirizzo sorg
;
; *********************************************************************************************

.macro     PushB    sorg
           lda      sorg                    ;preleva il byte
           pha                              ;lo memorizza nello stack
.endm
```

```
; ****************************************************************************************
;
;  Push Word:       PushW sorg
;
;  Argomento:       sorg                      indirizzo della word da inserire
;                                             nello stack
;
;  Funzione:        Memorizza nello stack la word all'indirizzo
;                   sorg
;
; ****************************************************************************************

.macro  PushW       sorg
        lda         sorg + 1                 ;preleva il byte alto della word
        pha                                  ;lo inserisce nello stack
        lda         sorg + 0                 ;preleva il byte basso della word
        pha                                  ;lo inserisce nello stack
.endm


; ****************************************************************************************
;
;  Pop Byte:        PopB dest
;
;  Argomenti:       dest                      locazione alla quale memorizzare il valore
;                                             prelevato dallo stack
;
;  Funzione:        Preleva un byte dallo stack
;
; ****************************************************************************************

.macro  PopB        dest
        pla                                  ;preleva il byte dallo stack
        sta         dest                     ;e lo salva
.endm
```

```
; ****************************************************************************************
;
;  Pop Word:        PopW dest
;
;  Argomenti:       dest                       locazione alla quale memorizzare
                                               la word prelevata dallo stack
;
;  Funzione:        Preleva una word dallo stack
;
; ****************************************************************************************

.macro  PopW        dest
        pla                                 ;preleva il byte basso della word
        sta         dest + 0                ;e lo memorizza
        pla                                 ;preleva il byte alto della word
        sta         dest + 1                ;e lo memorizza
.endm


; ****************************************************************************************
;
;  Branch Relative Always:   bra addr
;
;  Argomenti:       addr                       indirizzo al quale spostare il controllo
;
;  Funzione:        Effettua un branch incondizionato. Rispetto all'istruzione jmp, questa
;                   macro presenta il vantaggio di essere perfettamente rilocabile
;
; ****************************************************************************************

.macro  bra         addr
        clv
        bvc         addr
.endm
```

```
; *****************************************************************************************
;
;  Set Bit:          smb bitNumero, dest
;
;  Argomenti:        bitNumero              numero del bit che deve essere impostato a 1 (7 per
;                                           MSD, 0 per LSD)
;                    dest                   indirizzo del byte che contiene il bit da
;                                           impostare a 1
;
;  Funzione:         Imposta a 1 il bit indicato. La versione piu' veloce (smbf) non preserva
;                    l'accumulatore
;
; *****************************************************************************************

.macro  smb          bitNumero, dest
        pha
        lda          #(1<<bitNumero)
        ora          dest
        sta          dest
        pla
.endm

.macro  smbf         bitNumero, dest
        lda          #(1<<bitNumero)
        ora          dest
        sta          dest
.endm


; *****************************************************************************************
;
;  Reset Bit:        rmb bitNumero, dest
;
;  Argomenti:        bitNumero              numero del bit che dev'essere impostato a 0 (7 per
;                                           MSD, 0 per LSD)
;                    dest                   indirizzo del byte che contiene il bit da
;                                           impostare a 0
;
;  Funzione:         Imposta a 0 il bit indicato. La versione veloce (rmbf) non preserva
;                    l'accumulatore
;
; *****************************************************************************************
```

```
.macro  rmb        bitNumero, dest
        pha
        lda        #(1<<bitNumero)
        and        dest
        sta        dest
        pla
.endm

.macro  rmbf       BitNumero, dest
        lda        #(1<<bitNumero)
        and        dest
        sta        dest
.endm

; *****************************************************************************************
;
;  Branch on Bit Set:                  bbs bitNumero, sorg, addr
;
;  Argomenti:      bitNumero            numero del bit del byte che dev'essere
;                                       controllato (7 per MSD, 0 per LSD)
;                  sorg                 indirizzo del byte che contiene il bit da
;                                       controllare
;                  addr                 indirizzo al quale saltare se il bit e'
;                                       impostato a 1
;
;  Funzione:       Controlla il bit nel byte sorg, salta se e' impostato a 1. La versione
;                  veloce (bbsf) non preserva l'accumulatore
;
; *****************************************************************************************

.macro  bbs        bitNumero, sorg, addr
        php
        pha
        lda        sorg
        and        #(1<<bitNumero)
        beq        noBranch
        pla
        plp
        bra        addr
noBranch:
        pla
        plp
.endm
```

```
.macro  bbsf        bitNumero, sorg, addr
.if (bitNumero = 7)
        bit         sorg
        bmi         addr
.elif (bitNumero = 6)
        bit         sorg
        bvs         addr
.else
        lda         sorg
        and         #(1<<bitNumber)
        bne         addr
.endif
.endm


; ******************************************************************************************
;
;  Branch on Bit Reset:  bbr bitNumero, sorg, addr
;
;  Argomenti:       bitNumero           numero del bit del byte che dev'essere controllato
;                                       (7 per MSD, 0 per LSD)
;                   sorg                indirizzo del byte che contiene il bit da
;                                       controllare
;                   addr                indirizzo al quale saltare se il bit e'
;                                       impostato a 0
;
;  Funzione:        Controlla il bit nel byte sorg, salta se e' impostato a 0. La versione
;                   veloce (bbrf) non preserva l'accumulatore
;
; ******************************************************************************************

.macro  bbr         bitNumero, sorg, addr
        php
        pha
        lda         sorg
        and         #(1<<bitNumero)
        bne         noBranch
        pla
        plp
        bra         addr
noBranch:
        pla
```

```
        plp
.endm

.macro  bbrf        bitNumero, sorg, addr
.if                 (bitNumero = 7)
        bit         sorg
        bpl         addr
.elif               (bitNumero = 6)
        bit         sorg
        bit         addr
.else
        lda         sorg
        and         #(1<<bitNumero)
        beq         addr
.endif
.endm


; *****************************************************************************************
;
;  Increment Word:  IncW addr
;
;  Argomenti:       addr                    indirizzo della word da incrementare
;
;  Funzione:        IncW incrementa una word, e restituisce il flag zero (Z) impostato a 1
;                   se la word vale 0
;
; *****************************************************************************************

.macro  IncW        addr
        inc         addr                ;incrementa il byte basso della word
        bne         done                ;se diverso da zero il byte alto va bene
        inc         addr + 1            ;altrimenti incrementa il byte alto
done:
.endm
```

```
; ***************************************************************************************************
;
;  Negate Word:      NegateW wordaddr
;
;  Argomenti:        wordaddr              indirizzo della word che dev'essere negata
;
;  Funzione:         Nega la word puntata da wordaddr
;
; ***************************************************************************************************

.macro  NegateW      wordaddr
        ldx          #wordaddr            ;passa l'indirizzo della word da negare
        jsr          Dnegate              ;chiama la routine del Kernel
.endm
```

APPENDICE E: I FORMATI DEI FILE

Una panoramica

In questa appendice si descrive il formato dei dati di alcuni particolari file (Text Scrap, Photo Scrap e tutti quelli generati da geoWrite). I file Text Scrap e Photo Scrap sono stati realizzati per consentire alle applicazioni il reciproco scambio di dati grafici e testi. Vediamo brevemente la loro utilità. Quando l'utente attiva l'operazione cut (taglia) o copy (copia), durante l'esecuzione di un'applicazione, un file Photo Scrap o Text Scrap viene opportunamente creato su disco. Questo file è del tipo SYSTEM ed è a struttura SEQUENTIAL. L'utente può quindi mandare in esecuzione un'altra applicazione e attivare l'operazione paste (incolla) per trasferire i dati contenuti nel file Scrap all'interno del nuovo documento. I file Scrap possono essere anche collezionati in "album" tramite i desk accessory Photo Manager e Text Manager.

Il formato di output su disco adottato da geoWrite è importante per i programmatori che desiderano realizzare applicazioni in grado di creare file in quel formato, o leggere i documenti di geoWrite.

La prima parte di questa appendice illustra il formato per i file Photo Scrap; la seconda parte il formato per i file Text Scrap; la terza parte il formato utilizzato da geoWrite.

Versioni future

I formati per i file Photo Scrap e Text Scrap si sono già evoluti rispetto alle prime versioni, e probabilmente subiranno altri miglioramenti in futuro. Per evitare problemi di compatibilità, le applicazioni devono controllare la versione del formato, memorizzata nel File Header del file, prima di procedere all'elaborazione dei dati. Il

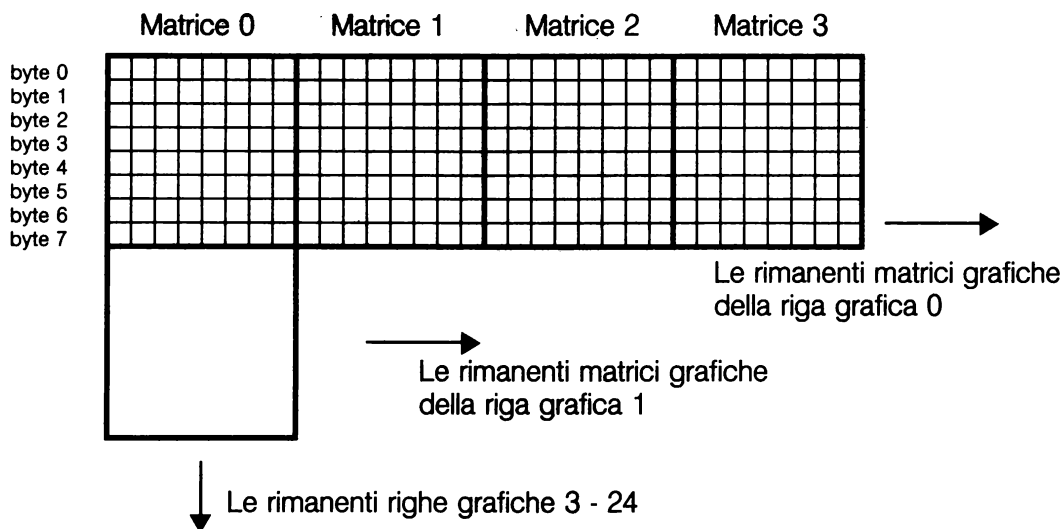
controllo della stringa versione di un file è ampiamente discusso nel capitolo dedicato al sistema dei file in ambiente GEOS. I byte in posizione 89 - 92 (decimali) del blocco File Header, contengono la stringa ASCII della versione, per esempio V1.1 o successive. La versione 1.1 indica il primo formato generale che è stato impiegato per i file di dati.

Se il file Scrap è in un formato meno recente di quelli che l'applicazione è in grado d'interpretare, dev'essere convertito. In genere le applicazioni dovrebbero essere in grado di fare questa conversione, ed essere quindi compatibili con tutti i file dati creati in precedenza. Se il formato dei dati contenuti nel file è più recente di quello adottato dall'applicazione, questa dovrebbe rinunciare all'elaborazione dei dati.

Photo Scrap

Attualmente i file Photo Scrap sono in grado di contenere una sola mappa grafica in bit-map. Una mappa grafica è una struttura che GEOS utilizza per contenere dati grafici in bit-map compattati. La compattazione dei dati grafici e le mappe grafiche sono argomenti che abbiamo affrontato nel capitolo 4. Il file Photo Scrap non è altro che una mappa grafica in bit-map, con allegata una tavola di colori.

Nel formato ottenuto dopo l'espansione dei dati, ogni byte della tavola di colori contiene informazioni sul colore di ogni matrice grafica di cui è composta la mappa scompattata. La tavola di colori si ottiene accedendo al buffer da 1000 byte che GEOS usa per i colori delle matrici grafiche per lo schermo in alta risoluzione. Come abbiamo già spiegato nel corso del libro, con il termine "matrice grafica" intendiamo una matrice di 8 x 8 pixel (quando non altrimenti specificato) che dev'essere allineata con gli spazi carattere dello schermo in bassa risoluzione.

Nel modo in alta risoluzione bit-map del C-64, una matrice grafica occupa 8 byte e definisce un quadrato di 8 x 8 pixel. A ogni matrice è associato un byte che ne definisce il colore. Per esempio, il primo byte della pagina di colori determina il colore della prima matrice grafica sullo schermo. Il secondo byte determina il colore della seconda matrice grafica, e così via. Nella tavola della pagina successiva è illustrata questa corrispondenza.

Organizzazione dei byte su uno schermo in bit-map

Le mappe grafiche memorizzate nei file Photo Scrap non sono limitate alle dimensioni dello schermo. Mentre la maggior parte delle applicazioni creano file Scrap di dimensioni inferiori a quelle dello schermo, possono ugualmente essere create e salvate mappa grafiche anche molto più grandi. La tavola di colori e la mappa grafica possono essere indifferentemente più piccole o più grandi dello schermo in alta risoluzione.

Il formato dei dati per un file Photo Scrap prevede che i primi tre byte che precedono il primo gruppo grafico individuino le dimensioni della mappa grafica. Più precisamente, il primo byte indica la larghezza della mappa grafica in byte, e i due byte successivi indicano l'altezza espressa in linee di scansione. Moltiplicando tra loro le due dimensioni si ottiene l'esatto numero di byte generato dall'espansione di tutti i gruppi grafici. L'altezza dev'essere sempre un multiplo di 8, dal momento che le operazioni cut e copy hanno effetto solo su matrici grafiche intere. La larghezza della mappa è sempre espressa in matrici complete. Queste restrizioni sono necessarie perché ogni byte del colore corrisponde a un'intera matrice grafica.

La tavola dei colori viene compattata come le mappe grafiche. Quindi anch'essa appare nel file Photo Scrap come una serie di gruppi grafici da espandere. Il primo gruppo grafico della tavola dei colori inizia subito dopo l'ultimo gruppo grafico della mappa in bit-map. Quando il previsto numero di byte grafici è stato espanso, il gruppo grafico successivo corrisponde all'inizio della tavola dei colori compattata. Il numero di byte della mappa grafica diviso per 8 individua il numero di byte di cui è composta la tavola dei colori. La tavola della pagina successiva illustra la struttura di un file Photo Scrap.

Formato dei dati in un file Photo Scrap

Byte	Contenuto
0	Larghezza in byte della mappa grafica in bit-map
1-2	Altezza in linee di scansione della mappa grafica in bit-map
3	Byte di conteggio del primo gruppo grafico. GEOS prevede tre diversi formati per i gruppi grafici. La loro attivazione dipende dal valore contenuto nel byte di conteggio: 0-127 ripeti il prossimo byte COUNT volte 128-220 scrivi una sola volta, nell'ordine, i seguenti (COUNT-128) byte (straight bit-map) 221-255 utilizza il byte che segue come BIGCOUNT, e ripete i seguenti (COUNT-220) byte BIGCOUNT volte
4-fine gruppo grafico	Dati del gruppo grafico in uno dei tre metodi di compattazione
Count	Byte di conteggio del successivo gruppo grafico
Dati bit-map	Dati del gruppo grafico
Altri gruppi grafici	
Tavola Colore	Tavola dei colori memorizzata in modo compattato

Per riassumere, un file Photo Scrap è composto dai tre byte delle due dimensioni, seguiti dalla mappa grafica compattata e dalla tavola dei colori. Sia la mappa grafica sia la tavola dei colori sono composte da gruppi grafici sequenziali. I gruppo grafici possono adottare uno qualsiasi dei tre formati di compattazione di cui dispone GEOS. Un gruppo grafico è composto dal byte di conteggio (Count Byte) che ne individua il formato di compattazione e il parametro di conteggio, seguito da una serie di byte che dipendono dal tipo di compattazione. Come è già stato descritto nel capitolo dedicato alla grafica, dopo l'espansione le mappe grafiche in bit-map devono essere riordinate in linee di scansione a matrici grafiche. La tavola dei colori contiene, in formato compatto, i colori della mappa grafica. Ogni byte della tavola individua il colore della corrispondente matrice grafica 8 x 8 pixel memorizzata nella mappa grafica.

Text Scrap V1.2

Questo paragrafo descrive il formato dei file Text Scrap V1.2. La versione 2.0 è un ampliamento della versione 1.2. L'unica vera differenza consiste nella riga di

definizione (ruler) presente nei file Photo Scrap V2.0, che sarà illustrata nei prossimi paragrafi.

Un file Text Scrap è una stringa ASCII contenente alcuni caratteri di controllo. I caratteri di controllo sono facilmente individuabili in quanto non sono stampabili come codici ASCII(†). Nei file Text Scrap sono utilizzati due particolari caratteri di controllo. Il primo è il TAB (codice $09). Sta alla particolare applicazione rendere disponibili i caratteri tabulatori. Il secondo è individuato dalla costante NEWCARDSET (23). Questa segnala all'applicazione l'inizio di una stringa di 4 byte per la gestione delle fonti e degli stili. I primi due byte dopo NEWCARDSET caratterizzano la fonte che dev'essere utilizzata per visualizzare il testo che segue. L'ultimo byte della stringa (style byte) indica lo stile che dev'essere impiegato: tondo (plaintext), nero (bold), corsivo (italic), sottolineato (underline) e outline. Ogni stile è controllato da un bit dello style byte. Impostando, per esempio, il bit "nero", il testo viene visualizzato con lo stile nero. Nella tavola successiva è illustrato il significato di ogni bit.

Nel testo appare una completa stringa NEWCARDSET ogni volta che si richiede un cambiamento nella fonte e/o nello stile. L'accessorio da scrivania Text Manager non visualizza le tabulazioni, i cambiamenti di stile e di fonte, ma questi caratteri di controllo sono ugualmente presenti nel file Text Scrap. Le applicazioni devono essere in grado di gestire anche i caratteri di controllo presenti all'interno di un file Text Scrap, oltre a quelli ASCII. Vediamo ora la struttura di un file Text Scrap.

Text Scrap

Il file Text Scrap, come appare in memoria, inizia con due byte che indicano il numero totale di byte che seguono (esclusi i primi due). Subito dopo segue una stringa NEWCARDSET.

La stringa di controllo è lunga quattro byte e inizia con il carattere di controllo NEWCARDSET. I due byte successivi costituiscono l'identificatore del set di caratteri selezionato (Font ID). I 6 bit più bassi di questa word specificano il corpo del carattere (point size). I 10 bit più alti specificano il numero di identificazione unico della fonte. La word del Font ID è seguita dallo style byte, all'interno del quale ogni bit controlla un particolare stile. Impostando un bit all'interno dello style byte, si attiva lo stile corrispondente. Cancellandolo, si ottiene la disattivazione dello stile. Se tutti i bit dello style byte sono azzerati, è attivato lo stile tondo (plain text). Per uno schematico quadro riassuntivo si può consultare la tavola della pagina successiva.

(†) In ASCII il set di caratteri stampabili inizia con il carattere spazio (SPACE) al quale corrisponde il codice $20. I primi 32 caratteri ASCII (dal codice $00 al codice $1F) non sono stampabili in quanto non corrispondono né a lettere né a numeri, e spesso vengono impiegati come caratteri di controllo all'interno delle stringhe di testo.

Formato dei file Text Scrap

Byte	Contenuto	Commento		
0-1	Lunghezza	Numero di byte che seguono nel file		
2	NEWCARDSET	Carattere di controllo per la selezione della fonte e dello stile (codice 23)		
3-4	Font ID	Indicatore della fonte e del set. I 6 bit piu' bassi specificano il corpo (il set), mentre i 10 piu' alti specificano l'identificatore unico della fonte che dev'essere attivata		
5	Style Byte	Costante	Valore	Funzione
		SET_UNDERLINE	%10000000	Bit 7 = 1: attiva sottolineato
		SET_BOLD	%01000000	Bit 6 = 1: attiva nero
		SET_REVERSE	%00100000	Bit 5 = 1: attiva negativo
		SET_ITALIC	%00010000	Bit 4 = 1 attiva corsivo
		SET_OUTLINE	%00001000	Bit 3 = 1: attiva outline
		SET_SUPERSCRIPT	%00000100	Bit 2 = 1: attiva gli indici all'apice
		SET_SUBSCRIPT	%00000010	Bit 1 = 1: attiva gli indici al pedice
		SET_PLAINTEXT	%00000000	Tutti i Bit a 0: attiva tondo
6-fine file	Stringa di testo	La stringa ASCII contenente i caratteri di controllo (TAB e NEWCARDSET), e, se V2.0, la riga di definizione (ruler)		

Il resto della stringa è composta da testo, caratteri TAB e altre eventuali stringhe NEWCARDSET. Non è previsto un particolare carattere che segnali la fine del testo, e quindi le applicazioni che accedono a file Scrap di testo devono confrontare il numero di caratteri letti con il totale indicato dai primi due byte del file.

La tavola che segue riassume schematicamente le fonti carattere attualmente disponibili su GEOS. Le fonti per geoLaser sono state realizzate in modo tale da riprodurre, per quanto possibile, i caratteri della stampante laser Apple LaserWriter.

Per riassumere, un file Text Scrap inizia con una word che indica la lunghezza del testo (word esclusa), seguita da una stringa di controllo iniziale (NEWCARDSET) per la selezione delle fonti e degli stili. Solo allora inizia il testo vero e proprio, che a sua volta può contenere il carattere di controllo TAB e le stringhe NEWCARDSET. Questo è il formato utilizzato nei file Text Scrap V1.2

Le fonti carattere di GEOS

Nome fonte	Identificatore (10 bit più alti)	Corpo (6 bit più bassi)	Word Font ID
BSW	0	9	$00 09
University	1	6	$00 46
		10	$00 4A
		12	$00 4C
		14	$00 4E
		18	$00 52
		24	$00 58
California	2	10	$00 8A
		12	$00 8C
		13	$00 8D
		14	$00 8E
		18	$00 92
Roma	3	9	$00 C9
		12	$00 CC
		18	$00 D2
		24	$00 D6
Dwinelle	4	18	$01 12
Cory	5	12	$01 4C
		13	$01 4D

Le fonti dalla 6 alla 25 risiedono nel disco GEOS Font Pack 1

Nome fonte	Identificatore (10 bit più alti)	Corpo (6 bit più bassi)	Word Font ID
Tolman	6	12	$01 8C
		24	$01 98
Bubble	7	24	$01 D8
FontKnox	8	24	$02 18
Harmon	9	10	$02 4A
		20	$02 54
Mykonos	10	12	$02 8C
		24	$02 98
Boalt	11	12	$02 CC
		24	$02 D8
Stadium	12	12	$02 30
Tilden	13	12	$03 CC
		24	$03 4C
Evans	14	18	$03 92
Durrant	15	10	$03 CA
		12	$03 CC
		18	$03 D2
		24	$03 D8
Telegraph	16	18	$04 12
Superb	17	24	$04 58
Bowditch	18	12	$04 8C
		24	$04 D8
Ormand	19	12	$04 CC
		24	$04 98
Elmwood	20	18	$05 12
		36	$05 24
Hearst	21	10	$05 4A
		12	$05 4C
		18	$05 52
		24	$05 58
Brennens	22	18	$05 92
Channing	23	14	$05 CE
		16	$05 D0
		24	$05 D8
Putnam	24	12	$06 0C
		24	$06 18
LeConte	25	12	$06 4C
		18	$06 52

Fonti per geoLaser

Nome fonte	Identificatore (10 bit più alti)	Corpo (6 bit più bassi)	Word Font ID
Commodore	26	10	$06 8A
LW_Roma	27	9	$06 C9
		10	$06 CA
		12	$06 CC
		14	$06 CE
		18	$06 D2
		24	$06 D8
LW_Cal	28	9	$07 09
		10	$07 0A
		12	$07 0C
		14	$07 0E
		18	$07 12
		24	$07 18
LW_Greek	29	9	$07 49
		10	$07 4A
		12	$07 4C
		14	$07 4E
		18	$07 52
		24	$07 58
LW_Barrows	30	9	$07 89
		10	$07 8A
		12	$07 8C
		14	$07 8E
		18	$07 92
		24	$07 98

La riga di definizione V2.0

Nella versione 2.0 dei file Text Scrap è stata aggiunta la riga di definizione (ruler) del testo, per mantenere la compatibilità con il formato geoWrite. In particolare, con la versione 2.0 di geoWrite è diventato possibile inserire più funzioni di giustificazione e di cambio margini all'interno di una stessa pagina (cioè più righe di definizione all'interno della stessa pagina). La riga di definizione è una struttura invisibile all'utente, generata dall'applicazione per memorizzare lungo il testo parametri fondamentali come i margini, la presenza di tabulatori e così via. Questa riga non deve necessariamente apparire nel corso del testo, ma, se viene introdotta, la sua posizione

è condizionata a particolari aree. Più precisamente, la riga di definizione può apparire solo all'inizio del file o all'inizio di un paragrafo. I paragrafi sono individuati nel testo dalla presenza dei caratteri CR (ritorno carrello), e quindi una riga grafica dev'essere sempre anteposta al carattere CR che termina il paragrafo precedente. Le righe grafiche sono lunghe 27 byte, e contengono informazioni sui margini del documento, la giustificazione dei paragrafi, i colori (se disponibili) e così via.

Anche se nelle righe di definizione del testo sono specificate le tabulazioni, l'accessorio da scrivania Text Manager non le interpreta. Se l'applicazione prevede le tabulazioni, quando incontra nel testo il carattere TAB deve spostare il cursore alla colonna di tabulazione specificata dall'ultima riga di definizione incontrata. Se a destra del cursore non è stata impostata nessuna tabulazione, il cursore viene spostato a riga nuova. La tavola che segue illustra il formato V2.0 della riga di definizione.

Formato V2.0 della riga di definizione

Byte	Contenuto	Descrizione	
1	ESC_RULER	Indica l'inizio della riga di definizione	
2 - 3	Left Margin	Margine sinistro espresso in pixel (0 - 319)	
4 - 5	Right Margin	Margine destro espresso in pixel (Left Margin<Right Margin≤319)	
6 - 21	Tabulatori	Ogni tabulazione occupa una word, 8 word totali per 8 tabulatori	
		Bit 15	Impostato a 1 tabulatore decimale, allineamento decimale dei punti Impostato a 0 tabulatore normale nel testo
		Bit 14-0	Posizione del tabulatore (tabulatore<Right Margin)
22 - 23	Rientranza	Distanza dal margine sinistro da impostare all'inizio di un nuovo paragrafo (0 - 319)	
24	Giustificazione	Bit per la giustificazione e l'interlinea	
		Bit 1 Bit 0	
		0 0	testo allineato a sinistra
		0 1	testo centrato
		1 0	testo allineato a destra
		1 1	testo allineato a sinistra e a destra

SEGUE

SEGUE

		Bit 3 Bit 2	
		0 0	interlinea di 1 riga
		0 1	interlinea di 1,5 righe
		1 0	interlinea di 2 righe
25	Colore del testo	Colore del testo. Attualmente nessuna applicazione GEOS utilizza questo byte	
26 - 27	Riservati	Byte riservati per impieghi futuri	

Formato dei file generati da geoWrite

Come avviene per i file Text Scrap, anche per i file generati da geoWrite esistono due formati: il formato V1.1 e il formato V2.0. I numeri di versione dei formati di output non corrispondono ai numeri delle varie versioni di geoWrite. Mentre per geoWrite esistono le versioni 1.1, 1.2, 1.3 e 2.0, i formati di output hanno solo la versione 1.1 e la 2.0.

In entrambi i formati di output, i documenti sono memorizzati in file a struttura VLIR. In generale, ogni record del file contiene i dati di una pagina di testo. Alcuni record sono utilizzati per le figure grafiche e, nel caso dei file in formato V2.0, per le informazioni relative alla nota di fine pagina e all'intestazione. La tavola che segue mostra l'organizzazione complessiva di questa struttura.

Formato dei file VLIR generati da geoWrite

Numero del record	**Formato V1.1 dei File**	**Formato V2.0 dei File**
0 - 60	Pagine di testo	Pagine di testo
61	Pagine di testo	Intestazione, vuoto in assenza di testo
62	Pagine di testo	Nota a fine pagina, vuoto in assenza di testo
63	Pagine di testo	Riservato
64 - 127	Figure in formato BitmapUp	Figure in formato BitmapUp

La principale differenza fra il formato V1.1 e il formato V2.0 consiste nella gestione dell'intestazione e della nota a fine pagina messe a disposizione da Writer's Workshop V2.0. Le pagine 61 - 63 possono essere impiegate per memorizzare testi, come avviene con geoWrite V1.2 e V1.3, ma se il documento viene in seguito modificato con geoWrite V2.0, queste pagine non vengono considerate. Questo limite nella compatibilità tra i due formati non dovrebbe comunque costituire un problema, soprattutto se si considera che nessuno ha mai memorizzato un file testo di 61 pagine in un disco del 1541. Quando sarà disponibile la gestione della doppia faccia sul 1571, questo problema potrebbe ripresentarsi.

In geoWrite, ogni documento è suddiviso in pagine separate, e ogni pagina viene memorizzata in un record della struttura VLIR del file. Una pagina consiste in una riga di definizione seguita dal testo. Nei file in formato V1.1, prodotti dalle versioni 1.2 e 1.3 di geoWrite, la riga di definizione include soltanto l'indicazione dei margini e dei tabulatori.

Formato V1.1 di una pagina di geoWrite

Byte	**Contenuto**	**Descrizione**
0 - 1	Left Margin	Coordinata in pixel del margine sinistro
2 - 3	Right Margin	Coordinata in pixel del margine destro
4 - 19	Tabulatori	Array da 8 word ognuna delle quali contiene una coordinata in pixel del tabulatore
20 - 23	NEWCARDSET	Stringa di selezione delle fonti e degli stili
24 - ?	Testo e Grafica	Testo della pagina. Puo' contenere cambi di fonte e di stile, indicatori di inserimenti grafici e identificatori di fine pagina PAGE_BREAK = 1 geoWrite considera conclusa la pagina corrente e inizia a scrivere in quella successiva ESC_GRAPHICS = 16 Carattere di controllo per l'inserimento di una figura
Ultimo byte	EOF = 1	Marcatore di fine pagina

Il testo che segue la prima stringa NEWCARDSET è memorizzato in ASCII. Le stringhe di controllo vengono utilizzate per i cambi di fonte e di stile, e per includere le figure grafiche. I dati grafici di ogni figura sono memorizzati in record separati. Tutte le pagine non vuote devono iniziare con la stringa di controllo NEWCARDSET. Due stringhe di questo tipo non possono essere consecutive.

All'interno di un testo possono essere presenti alcune figure: geoWrite segnala il punto d'inserimento tramite il carattere ESC_GRAPHICS, e memorizza i dati della figura, sotto forma di mappa grafica, in un particolare record. Per informazioni più dettagliate sulle mappe grafiche in ambiente GEOS si consulti il capitolo 4.

Stringa grafica di controllo

Byte	Funzione	Descrizione
0	ESC_GRAPHICS	Carattere che segnala la presenza di un stringa di controllo (codice $10)
1	Larghezza	Larghezza del disegno in matrici grafiche
2 - 3	Altezza	Altezza del disegno in linee di scansione
5	Numero del record	Numero del record contenente i dati grafici del disegno inserito nel testo. Il disegno dev'essere stato prelevato da un file Photo Scrap

geoWrite V2.0

La versione 2.0 di geoWrite, dal punto di vista dell'output su file, è simile alla versione 1.2, ma mette a disposizione una riga di definizione più ricca di informazioni, con una struttura analoga a quella utilizzata nei file Text Scrap. Il formato di una pagina generata da geoWrite V2.0 è illustrato nella tavola che segue.

Formato della pagina generata da geoWrite V2.0

Byte	Descrizione	
0 - 27	Stringa della riga di definizione V2.0 del testo	
28 - 31	NEWCARDSET (23) per il cambio della fonte e dello stile	
32 - ?	Testo del documento. Puo' contenere altre righe di definizione, cambi di fonte e di stile, disegni e interruzioni di pagina	
	PAGE_BREAK = 1	Conclude la pagina corrente e ne inizia una nuova
	ESC_GRAPHICS = 16	Inizio di un disegno
Ultimo byte	EOF = 0	Marcatore di fine file

Nel formato V2.0 ulteriori informazioni sono immagazzinate nel blocco File Header del file di testo. Per esempio l'altezza dell'intestazione e della nota a fine pagina, l'altezza di una pagina del testo (questo valore dipende dal tipo di stampante adottata), e alcuni flag per il modo NLQ e la titolazione.

Informazioni nel File Header dei file V2.0

Byte	Contenuto	Descrizione
137 - 138	Numero prima pagina	Numero che si assegna alla prima pagina (non dev'essere necessariamente 1)
139	Titolo e NLQ	Bit 7 impostato a 1: la prima pagina contiene il titolo e dev'essere stampata senza intestazione ne' nota di fine pagina Bit 6 impostato a 1: la stampa puo' essere effettuata solo in modo NLQ
140 - 141	Altezza dell'intestazione	Altezza in pixel che dev'essere riservata su ogni pagina per l'intestazione
142 - 143	Altezza della nota di fine pagina	Altezza in pixel che dev'essere riservata su ogni pagina per la nota di fine pagina
144 - 145	Altezza della pagina	Stampanti diverse supportano diverse risoluzioni verticali. Se l'altezza della pagina memorizzata in questa word non corrisponde a quella imposta dalla stampante, geoWrite V2.0 riformatta il file per renderlo compatibile

Riepilogo su geoWrite

I documenti generati da geoWrite vengono salvati in file a struttura VLIR. Ogni pagina viene memorizzata su disco all'interno di un opportuno record. Alcuni record del file possono anche contenere i dati per i disegni da inserire nel testo. Oltre a questo, geoWrite V2.0 aggiunge la possibilità di definire un'intestazione, una nota a fine pagina, il tipo di giustificazione, il tipo d'interlinea, il modo NLQ e la pagina di copertina priva d'intestazioni. Nella versione 1.1 dei file la riga di definizione ha una struttura relativamente semplice e può trovarsi solo all'inizio della pagina. Nella versione 2.0 la riga grafica può essere inserita all'inizio di ogni paragrafo e contiene un numero maggiore di informazioni.

Questa panoramica sulle caratteristiche di geoWrite nelle sue diverse versioni dovrebbe essere sufficiente per leggere e creare file in uno qualsiasi dei formati disponibili. I programmatori, durante la realizzazione di un'applicazione, devono sempre tener presente che le nuove versioni sono espansioni delle precedenti e quindi non creano problemi di compatibilità. Per esempio, i file Text Scrap V1.1 possono essere letti dall'ultima versione di Text Manager. L'unica possibile incompatibilità fra i formati può verificarsi nel caso che geoWrite V1.2, o V1.3, salvi su disco un file tanto grande (più di 60 pagine) da occupare i record che geoWrite V2.0 utilizza per l'intestazione e la nota a fine pagina. Ma questa condizione non dovrebbe verificarsi mai, dal momento che è improbabile che un documento di oltre 60 pagine venga salvato su un solo disco.

I file Text Scrap e geoWrite differiscono soltanto per quanto riguarda la lunghezza: i file Text Scrap vengono salvati in file SEQUENTIAL e quindi al massimo possono raggiungere la lunghezza di una pagina. Si prestano quindi a essere utilizzati per la comunicazione tra due word processor.

INDICI

Indice delle routine

Per argomenti

Testi

Alfabetico

Indice delle variabili

Per argomenti

Alfabetico

Indice analitico

GEOS

Il nuovo sistema operativo per i computer Commodore 64/64c/128/128D

I computer a otto bit non saranno mai più gli stessi.

Scritta dai creatori di GEOS – i programmatori della Berkeley Softworks – la *Guida ufficiale alla programmazione di GEOS* è il testo indispensabile per conoscere a fondo i segreti di questo nuovo ambiente operativo e creare stupende applicazioni dotate d'interfaccia utente grafica, menu a scomparsa, icone, finestre, box di dialogo, fonti proporzionali di ogni corpo e forma, processi eseguibili in multitasking, tempi minimi d'accesso al disco, RAM disk, gestione in overlay delle applicazioni, controllo semplificato del mouse, driver di input e di stampa... tutte caratteristiche che fanno di GEOS un ambiente di lavoro e sviluppo davvero professionale.
La guida descrive tutte le routine di sistema (oltre 170), l'intera mappa di memoria, i driver di input e di stampa per i dispositivi più comuni, approfondisce tutti gli argomenti che si devono conoscere per programmare in ambiente GEOS e svela i trucchi per lavorare anche in ambiente GEOS 128.

In quest'opera si descrivono con ricchezza di particolari tutti gli aspetti di GEOS, senza trascurare alcun dettaglio utile per la programmazione. Scopriteli e fate conoscenza con:

- La gestione dei flussi di eventi
- Le icone e i menu
- La grafica, i testi e le fonti carattere
- I driver di input e di stampa
- I processi temporizzati, le capacità multitasking
- I box di dialogo
- Il sistema di gestione dei file e il turbo
- La libreria matematica
- GEOS 128, la grafica a 80 colonne
- Le espansioni RAM e il RAM disk

La *Guida ufficiale alla programmazione di GEOS* è la più completa fonte d'informazioni sul mondo di GEOS. L'edizione italiana è stata redatta apportando al testo originale aggiornamenti e correzioni, un lavoro che si è svolto a stretto contatto con i programmatori della Berkeley Softworks.

ISBN 88-7803-003-1

Lire 64.000

GUIDA UFFICIALE ALLA PROGRAMMAZIONE DI GEOS

IHT
GRUPPO EDITORIALE